Ingeniería de Telecomunicación

PROYECTO FIN DE CARRERA

Diseño y Validación del Control Digital de un Inversor de Potencia en Ejes de Referencia Síncronos Conectado a Red

Autor: Beatriz Brogeras García

Tutores: Celia López Ongil
Carlos Lucena Fernández

Leganés, Octubre de 2011

Título: Diseño y Validación del Control Digital de un Inversor de Potencia en Ejes de Referencia Síncronos Conectado a Red
Autor: Beatriz Brogeras García
Director: Celia López Ongil
Codirector: Carlos Lucena Fernández

TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 20__ en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

A mi padre.

Agradecimientos

En primer lugar, querría mostrar mi más sincero agradecimiento a los tutores de este proyecto: a Carlos Lucena, por ofrecerme la oportunidad de llevar a cabo este trabajo, por su cordial colaboración y disponibilidad siempre que lo he necesitado y a Celia López Ongil, por su inestimable ayuda, consejos y palabras de ánimo tanto en el ámbito académico como en el personal.

A mis padres, quiero darles las gracias por haberlo dado todo por sus hijas, por creer en mí en todo momento, por enseñarme a crecer ante las situaciones difíciles y animarme a luchar por lo que quiero. Gracias a ellos y al continuo respaldo de mi hermana Cristina he conseguido llegar al final de esta etapa.

En especial, quiero dedicar este proyecto a mi padre, porque ha sido y será todo un referente en mi vida. Es imposible no echarte de menos cada día.

A María, quiero agradecerle el hacer estos años más alegres, su compañerismo y su amistad que va más allá de las aulas.

Gracias a Juanjo, Jose María y César por su amistad incondicional, su apoyo durante la carrera, y en definitiva, por estar siempre ahí.

Finalmente, gracias a todos mis amigos por todos los momentos vividos que han endulzado estos años, y como no, a mi pequeña segunda familia durante mi estancia en Alemania: Laura, Anita y Rober.

A todos ellos, Gracias.

La fuerza no proviene de la capacidad corporal, sino de una férrea voluntad.
Mahatma Gandhi.

Resumen

El aumento de los sistemas electrónicos de potencia conectados a la red eléctrica puede causar un impacto negativo en la misma, debido entre otras cosas a los armónicos de tensión que producen. Habitualmente, se emplean convertidores de potencia que consiguen eliminar los armónicos de alta frecuencia, logrando una tensión con baja distorsión armónica a su salida. No obstante, la tensión de salida de estos convertidores puede tener armónicos de baja frecuencia debido a diferencias entre los interruptores de los convertidores o los tiempos muertos necesarios para evitar cortocircuitos. Con el fin de eliminar estos armónicos, se pueden usar esquemas de control en lazo cerrado para disminuir aun más la distorsión de la tensión a la salida.

En este proyecto se realiza un estudio del diseño de control de corriente en lazo cerrado, utilizando un sistema de referencia síncrono, en el que los ejes giran con la pulsación del sistema, obteniendo variables de control contínuas. De esta forma, la acción de control queda simplificada al ser todas las variables magnitudes continuas en régimen permanente. Aunque la implantación de estos controles resulta complicada, se pueden diseñar controles PI para anular el error en régimen permanente entre la referencia y la salida.

El control de los convertidores de potencia se ha llevado a cabo tradicionalmente usando técnicas analógicas, no obstante, actualmente existe una tendencia a sustituir dichos controladores analógicos por otros digitales que precisan un previo muestreo de las señales mediante dispositivos de conversión analógico-digital.

A lo largo de este Proyecto Final de Carrera se abordará el diseño y la validación del control digital de convertidores, realizando el desarrollo del circuito en un lenguaje de descripción hardware, de forma que se aúnen las ventajas de portabilidad y optimización del diseño en la aplicación propuesta. El proceso de validación del circuito de control digital implica una cosimulación analógico-digital que se apoya en dos de los simuladores más extendidos actualmente en el campo del diseño hardware.

Palabras clave: Inversores de potencia, eliminación selectiva de armónicos, control en lazo cerrado, tensión de salida de un inversor, conversión analógico-digital, diseño digital, lenguaje de descripción hardware, validación mixta.

Abstract

The increase of power electronic systems connected to grid can have a negative impact on it, due to the voltage harmonics produced. Typically, power converters are used in order to manage the cancelation of high-frequency harmonics, achieving a voltage with low harmonic distortion at the output of the converters. However, the output voltage of these inverters can have low frequency harmonics due to differences between switches or due to the dead time needed to avoid short-circuits. In order to cancel these harmonics, closed loop control schemes can be used, getting a decrease on the output voltage distortion.

In this project, a closed loop current control design is done, by using a synchronous reference system where the axis rotates with the system, getting continuous control variables. Thus, the control action is simplified due to each variable in steady state takes continuous magnitude. Although the implantation of these controls is difficult, PI controls can be designed to override the error in steady state between the reference and the output.

Control of power converters has been traditionally carried out using analog techniques, however, nowadays there is a tendency to replace them by digital ones, that requires a previous digital sampling of the signals by using analog-digital conversion devices.

Along this Final Project will be developed the design and validation of digital control converters, and subsequently, the circuit will be developed in a hardware description language, to unify the advantages of portability and design optimization in the proposed application. The validation process of the digital control circuit implies an analog-digital cosimulation supported by two of more extended simulators on the hardware design frame.

Keywords: Power inverter, selective harmonic cancelation, closed loop control, output voltage of an inverter, analog-digital conversion, digital design, hardware description language, mixed validation.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este primer capítulo se muestra la motivación del Proyecto Fin de Carrera, se establecen los objetivos fijados para el mismo y se realiza una descripción de los aspectos generales relativos a las fases de desarrollo, su planificación temporal y los medios que han sido necesarios para su realización. Por último, se muestra la estructura de capítulos que componen este Proyecto.

1.1 Justificación

La motivación de este proyecto parte de la tendencia actual en el diseño de convertidores de potencia, que deriva en una sustitución de los clásicos convertidores analógicos por otros digitales. Esto se debe al aumento de prestaciones que viene implícito a la digitalización.

La evolución en prestaciones de la tecnología digital permite resolver problemas complejos inviables para la tecnología analógica, además de proporcionar flexibilidad a la hora de posibles reconfiguraciones, ya que dichas reconfiguraciones no implicarían rediseño hardware con la consecuente comprobación y verificación del correcto funcionamiento como en el caso de tecnologías analógicas.
También importante, es el papel que desempeña la precisión, cuyo control de requisitos en una tecnología digital proviene de la especificación en la precisión del conversor analógico-digital (A/D) y del procesador digital de señales, en términos de longitud de

palabra, aritmética de coma flotante frente a coma fija y otros factores similares. Sin embargo, las tolerancias en los componentes de los circuitos analógicos, hacen que para el diseñador se complique el control de precisión de un sistema de procesado analógico.

Por otro lado, la evolución en prestaciones y precio que ha experimentado la electrónica digital **[1]** en las últimas décadas, hace que, en algunos casos, la implementación digital del sistema de procesado de señales sea más barato que su equivalente analógico, debido al menor coste del hardware digital o incluso al resultado de la flexibilidad ante modificaciones que permite la implementación digital.

Sin embargo, la implementación digital también tiene sus limitaciones. Una limitación práctica es la velocidad de operación de los conversores A/D y de los procesadores digitales de señales. Las señales con anchos de banda extremadamente grandes precisan conversores A/D con una velocidad de muestreo alta y procesadores digitales de señales rápidos.

Así pues, surge la necesidad de realizar simulaciones conjuntas de circuitos analógico-digitales en el ámbito de los circuitos electrónicos de potencia con control digital.

Debido a la carencia de herramientas informáticas que realicen simulaciones conjuntas aunando la potencia de las que las realizan en el mundo digital y el mundo analógico por separado, en este proyecto se alternarán la simulación analógica y la digital.

La electrónica de potencia es la rama de la ingeniería eléctrica, que haciendo uso de componentes electrónicos, procesos tecnológicos, teoría de circuitos y herramientas analíticas, consigue adaptar, controlar y transformar la energía eléctrica, con la finalidad de procesar la energía de la manera más eficiente posible.

Buscando este objetivo, se evita el uso de elementos resistivos en los circuitos, por ser potenciales generadores de pérdidas por efecto Joule, siendo los principales dispositivos empleados, los condensadores, las bobinas y los semiconductores en modo corte/saturación.

Atendiendo a las formas de energía que convierten, los circuitos de electrónica de potencia se clasifican en los siguientes grupos **[2]**:

- **Convertidor de corriente alterna a corriente alterna (c.a.-c.a.)**
  Permiten modificar el valor eficaz de la señal entregada a la carga por una fuente de corriente alterna, bien variando la frecuencia (ciclo convertidor), o bien sin alterarla (regulador de alterna).
- **Convertidor de corriente continua a corriente continua (c.c.-c.c.)**
  También denominado *"chopper"*, que permite suministrar una señal continua a la carga a partir de una alimentación de corriente continua.
- **Convertidor de corriente alterna a corriente continua (c.a.-c.c.) o rectificadores.**
  Pueden ser de tres tipos:

  - ✓ Rectificadores no controlados, constituidos por diodos y que no regulan la tensión de salida. Dicha tensión de salida siempre será positiva.

- ✓ Rectificadores semicontrolados, formados por diodos y por tiristores, que regulan la tensión de salida en magnitud, pero no en polaridad. La tensión de salida siempre es mayor o igual a cero.
- ✓ Rectificadores controlados, que emplean tiristores y que regulan la tensión de salida en magnitud y polaridad controlando el momento de disparo de los tiristores. La tensión de salida puede ser menor, igual o mayor que cero. Ya que estos convertidores permiten controlar el sentido de la potencia transferida a la carga, pueden funcionar como rectificadores o como inversores, y dado que el control se efectúa a través del ángulo de disparo de los tiristores, este tipo de convertidores recibe el nombre de convertidores controlados por fase.

- **Convertidores de corriente continua a corriente alterna (c.c.-c.a.)**

  También conocidos como inversores de frecuencia variable, que a partir de una alimentación de corriente continua proporcionan una corriente alterna de frecuencia regulable.

  A este último grupo pertenece el circuito de electrónica de potencia bajo estudio en este Proyecto Fin de Carrera.

Todas estas técnicas de conversión requieren de la conmutación de dispositivos semiconductores de potencia, que a su vez son controlados mediante señales generadas por circuitos integrados y componentes discretos, los cuales se han ido reemplazando por microprocesadores con el paso de los años.

En muchas aplicaciones industriales, muy a menudo es necesario controlar la tensión de salida de los inversores con el fin de hacer frente a las variaciones de la entrada continua (DC), para regular la tensión de los inversores y para cumplir los requisitos de tensión y frecuencia en la salida del circuito. Existen varias técnicas para modificar la tensión de salida de un inversor, siendo el método más común la incorporación en los inversores del control de modulación de ancho de pulso (PWM) **[3]**.

Este circuito de control gobernará generalmente el controlador o *"driver"* del interruptor, convirtiendo la señal que proporciona el circuito de control, en una señal con la energía y características necesarias para gobernar los interruptores en cada caso.

Así pues, un sistema de control debe ser capaz de lograr su objetivo, cumpliendo los siguientes requisitos:

- Garantizar la estabilidad y ser robusto frente a perturbaciones y errores en los modelos.
- Ser tan eficiente como sea posible, según un criterio preestablecido. Dicho criterio suele consistir en que la acción de control sobre las variables de entrada sea realizable, evitando comportamientos bruscos e irreales.
- Ser fácilmente implementable y cómodo de operar en tiempo real con ayuda de un circuito digital.

El control de los convertidores de potencia se ha llevado a cabo tradicionalmente usando técnicas analógicas, mediante el uso de circuitos integrados, diodos, transistores, redes resistivas, elementos capacitivos y comparadores.

No obstante, actualmente existe una tendencia a sustituir dichos controladores analógicos por otros digitales.

El uso de señales analógicas en un control digital requiere el previo muestreo de dichas señales, y para ello son necesarios dispositivos de conversión analógico-digital, de los que se obtienen palabras procesables por el control digital, con un número de bits determinado. Será una vez procesada esta información, cuando se generen las señales de control digitales.

Los sistemas de control digital presentan una serie de ventajas frente a los sistemas analógicos **[1]**:

- Capacidad de abordar problemas de control más complejos, incluso mejorando sus prestaciones.
- Menor susceptibilidad al deterioro debido al transcurso del tiempo o a factores del entorno.
- Presencia de componentes menos sensibles a los ruidos y a las vibraciones en las señales.
- Capacidad de integración con otros sistemas, estableciendo comunicación mediante protocolo con otro sistema, por ejemplo.
- Si se elige un dispositivo lógico programable se añade la ventaja de poder reprogramarlos, siendo fácil modificar el control para incluir nueva funcionalidad o corregir errores.
- Mayor fiabilidad y robustez del sistema.
- Mayor seguridad ante copia, ya que el encapsulado de los circuitos integrados es más seguro frente a la fácil lectura que exhiben por su naturaleza los sistemas analógicos tradicionales.

El control digital presenta ciertos aspectos de diseño a tener en cuenta ya que aumentan la complejidad del sistema:

- La necesidad del uso de convertidores analógico-digitales conduce a que se deba tener en cuenta la frecuencia de muestreo y la resolución en las señales muestreadas.
- La resolución efectiva de las señales y el desbordamiento en los cálculos deben ser considerados en el diseño.
- La resolución del ciclo de trabajo y la introducción de retardos también pueden plantear dificultades en el desarrollo del control.

A lo largo de este Proyecto Final de Carrera se abordará la solución del control digital de convertidores de potencia mediante el uso de dispositivos digitales de hardware específico o dedicado, en los cuales, la implementación física se pone particularmente al servicio de la tarea de control, permitiendo una concurrencia en la ejecución muy útil, al mismo tiempo que se optimiza el empleo de los recursos lógicos

Para esta clase de soluciones, se utilizan los circuitos digitales de aplicación específica (ASIC, *"Application Specific Integrated Circuit"*), que son circuitos integrados hechos a la medida de la aplicación y los circuitos programables de alta complejidad tales como CPLD (*"Complex Programmable Logic Device"*) o FPGAs (*"Field Programmable Gate Array"*) **[4]**.

Debido a la concurrencia con la que pueden realizar las operaciones estos dispositivos de hardware específico, será posible alcanzar una alta velocidad de operación al mejorar el tiempo de ejecución frente a una ejecución secuencial de instrucciones que tienen otros sistemas, como por ejemplo, los basados en microprocesador **[5]**.

Así pues, la concurrencia del hardware específico permite la maximización del uso de los recursos con estructuras en paralelo, en serie o segmentadas (*"pipeline"*).
Para aquellas operaciones muy sencillas de muestreo que hay que realizar, la mayoría de los sistemas basados en microprocesador no cumplen los requisitos de velocidad, mientras que un dispositivo de hardware especifico como puede ser una FPGA de cualquier gama, es capaz de llevar a cabo estas operaciones en el tiempo requerido.

Este tipo de soluciones mediante hardware específico son desarrolladas comúnmente con algún lenguaje de descripción del hardware, aunando así ventajas de portabilidad y optimización del diseño y permitiendo un diseño independiente de la tecnología al poder implementarse bien en circuitos programables o bien en un ASIC de cualquier fabricante.

La mayor parte de los avances de los últimos años en el sector industrial de los Circuitos Integrados de Aplicación Específica, han provisto un incremento de la velocidad y el rendimiento, así como una reducción notable en el tamaño y el consumo de los sistemas digitales, logrando el desarrollo de sistemas digitales cada vez más potentes y la generalización de estos sistemas en múltiples ámbitos de la industria y de la sociedad.

Desde su aparición, el diseño de circuitos integrados ha evolucionado de manera muy rápida, siendo motivo de este avance la utilidad de dichos dispositivos en los distintos campos que comprenden las tecnologías de la información, donde son capaces de desempeñar complejas funcionalidades con un bajo consumo de energía y grandes velocidades de ejecución, con un bajo precio en el mercado.

Debido a la creciente demanda de los circuitos integrados digitales, se ha trabajado profundamente en su mejora, llegando a lograr valiosos resultados, tanto en tecnologías de fabricación como en el proceso de diseño.

Actualmente, el proceso de diseño de un circuito integrado digital emplea gran número de herramientas de ayuda al diseño, CAD (*"Computer-Aided Design"*). Estas herramientas y la metodología seguida hacen posible la obtención de diseños más complejos funcionalmente, en menor tiempo y más baratos que los primeros circuitos integrados desarrollados.

Dichos logros, han sido posibles gracias a cambios fundamentales en la concepción del diseño de circuitos integrados digitales. Uno de estos cambios consiste en la descripción de los circuitos según su funcionalidad, a través de su comportamiento, permitiendo la descripción en niveles de abstracción superiores al de los esquemáticos de puertas lógicas, con el apoyo de los lenguajes de descripción hardware, y la síntesis lógica automática.

Los lenguajes de descripción de hardware han permitido describir los circuitos desde el punto de vista de su funcionalidad. Con la estandarización del lenguaje VHDL por el IEEE (*"Institute of Electrical and Electronics Engineers"*), se generalizó el uso de estos

lenguajes y se estableció un formato común a todos los centros de desarrollo para el diseño de circuitos **[6]**.

La síntesis lógica automática permite obtener una descripción estructural del diseño a partir de una descripción funcional en lenguaje HDL. El diseñador trabaja en un nivel de abstracción cercano a la descripción funcional del diseño, con lo que se abstrae de tareas de bajo nivel, como pueden ser la captura de esquemático o la simulación lógica. Por tanto, el diseñador trabaja actualmente más rápido y con menos probabilidad de cometer errores, consiguiendo tiempos y costes de desarrollo mejores, fomentando la rentabilidad y complejidad del diseño.

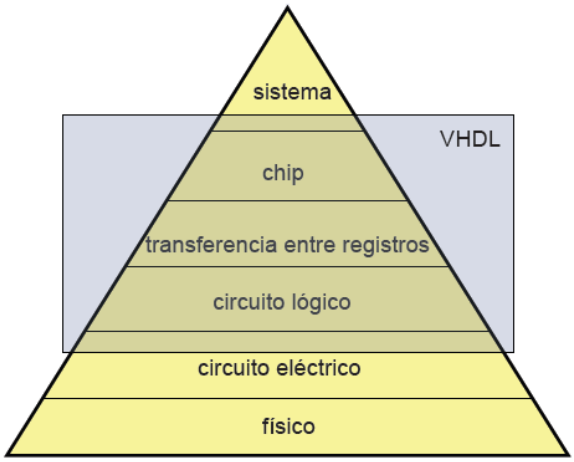


Figura 1.1. Niveles de diseño y dominios de representación

Como ya se ha dicho anteriormente, los circuitos integrados digitales se pueden implementar bien mediante un dispositivo programable (FPGA), o bien mediante un circuito integrado (ASIC).

Las FPGAs se caracterizan por su alta densidad de integración, su simplicidad en el uso y por su bajo coste de prototipado si se comparan con los ASICs. Si bien el coste unitario de los ASICs es muy competitivo, los costes fijos de fabricación y test son muy elevados, independientemente del número de unidades que se fabriquen Por este motivo, el uso de FPGAs representa una solución idónea para volúmenes de producción bajos/medios (hasta unas 30000 unidades), puesto que se aúnan las ventajas de la integración sin incurrir en los elevados costes de fabricación de los ASICs. Por lo tanto, los ASICs sólo compensan en coste en grandes tiradas frente a las FPGAs que, dada su difusión creciente, son cada vez más económicas y potentes.

No obstante, cuando una alta capacidad de cálculo es requerida, es necesario plantear un diseño basado en la repetición de operaciones aritméticas sencillas para sacar partido de la concurrencia, y este uso agota los recursos disponibles en las FPGAs.

En base a este criterio, el uso de ASICs permite explotar las características singulares del hardware específico, en especial la concurrencia y la velocidad de procesamiento, permitiendo a su vez un control que actúe a la frecuencia de conmutación (del orden de centenas de kHz).

De esta forma y a modo de conclusión, queda justificada la opción de implementar un regulador en un dispositivo digital de hardware específico mediante un lenguaje de descripción de hardware (VHDL).
Para concluir esta sección, a continuación se hace referencia a la modulación por ancho de pulso (PWM), por ser la técnica de control que será implementada a lo largo de este Proyecto Fin de Carrera.

La modulación por ancho de pulso es una técnica muy utilizada en el control de inversores trifásicos y consiste en modificar el ciclo de trabajo de una señal periódica, para controlar la cantidad de energía que envía el inversor a la red.

La construcción típica de un circuito PWM se lleva a cabo mediante comparadores de dos entradas, donde una de las entradas se conecta a una señal triangular que es la portadora y la otra a la señal sinusoidal moduladora. El ciclo de trabajo de la salida está en función de la portadora.

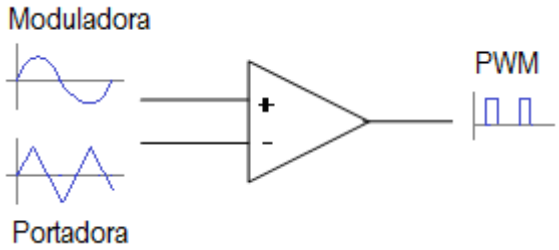


Figura 1.2. Circuito modulador.

Esta técnica es muy apropiada para implementación en medios digitales, donde una serie de transistores que funcionan como conmutadores, son accionados por señales digitales.

Más adelante, en el apartado Estado del Arte, se mostrará más en detalle la situación actual de los esquemas de control en sistemas eléctricos trifásicos. No obstante, puede adelantarse que, siendo dos de las técnicas más usadas en la simplificación de modelos matemáticos de sistemas trifásicos, las transformaciones de *Clarke* y *Park*, en este Proyecto Final de Carrera, se ha hecho uso de la transformación de *Park* que permite pasar a un sistema de referencia síncrono.

1.2 Objetivos

Tal y como se ha descrito en la sección anterior, el objetivo fundamental del Proyecto Final de Carrera es facilitar el diseño de convertidores de potencia con control digital. En los últimos años, esta solución está proliferando debido a las ventajas que presenta frente al control analógico tradicional.

Una vez elegidas las técnicas a usar para el diseño del inversor de Potencia, se procederá a su simulación hardware, para comprobar el correcto funcionamiento del inversor diseñado, y en base a dicho circuito implementado se realizará el diseño de la arquitectura, distinguiendo los bloques de control digital para su posterior implementación con un lenguaje de descripción de hardware como es VHDL.

De esta forma, con la implementación vía VHDL de nuestro sistema de control digital, podrá llegarse a un co diseño analógico-digital, permitiendo así una simulación menos abstracta y más cercana a la realidad y de más utilidad que la validación por separado de los diseños analógico y digital realizados por separado.

En base a este objetivo principal, se definieron los siguientes objetivos parciales:

- Estudio de los mecanismos disponibles en sistemas inversores con control digital.
- Análisis del circuito de control a implementar mediante VHDL y diseño de la arquitectura del circuito global.
- Estudio de los bloques de la arquitectura para su posterior implementación en VHDL y diseño de las rutas de datos para cada uno de los sub-bloques que compondrán el circuito de control del inversor.
- Implementación del control digital, para lo que será necesario:
  - Discretización e implementación de filtros de corriente.
  - Discretización e implementación del regulador PI.
- Simulación del circuito a implementar mediante PSIM y extracción de los datos de entrada y salida de cada uno de los bloques que compondrán el diseño digital en VHDL a ficheros que puedan ser tratados en Modelsim, para la posterior simulación de cada bloque.
- Depurado de cada uno de los bloques implementados, mediante la implementación de bancos de pruebas.
- Facilitar la simulación del sistema de control elegido, mediante el uso del lenguaje de descripción hardware VHDL que permitirá una integración del bloque de control digital en el circuito de potencia, siendo tratado como un sub-bloque digital a integrar en el resto del circuito analógico.

1.3 Fases de desarrollo

A continuación se establecen las fases de desarrollo y tareas en que se divide el proyecto, aplicando para ello los conocimientos adquiridos en la asignatura Proyectos de Ingeniería.

Este proyecto se ha desglosado en tres fases principales de desarrollo, tal y como se muestra a continuación:

- **Fase 1: Planificación:**

  - **Estudio de los sistemas inversores de potencia:** Realización de un estudio completo de los sistemas de control digitales disponibles y de las ventajas provistas por cada uno de ellos, analizando sus aplicaciones prácticas y revisando ejemplos existentes para poder poner en uso las posibilidades que ofrecen en el desarrollo de la aplicación propuesta.

- **Estudio del algoritmo de control elegido:** Análisis en profundidad del método de *Park* elegido para pasar a ejes de referencia síncronos, así como el estudio de reguladores PI y moduladores PWM para una mejor comprensión de las ventajas que aportan al sistema de control para inversores de potencia.

- **Estudio de las tecnologías necesarias:** Estudio del paquete de simulación PSIM y del lenguaje de diseño hardware VHDL, así como de la herramienta Modelsim como simulador VHDL. Implementación de ejemplos prácticos para la asimilación de estas tecnologías.

▪ **Fase 2: Desarrollo:**

- **Análisis y diseño inicial:** Elección de cada uno de los sub-bloques VHDL de que constará la implementación final que permita un desglose práctico a la hora de comprobar el correcto comportamiento de cada uno de ellos por separado, antes de ser integrados en el circuito final.

- **Implementación del sistema:** Desarrollo de todos los módulos y submódulos de que constará el sistema inversor, teniendo en cuenta el tipo de datos adecuado a considerar en el diseño sin perder viabilidad en el diseño de cada uno de los bloques.

- **Pruebas unitarias:** Creación de bancos de pruebas para verificar el diseño mediante simulación de cada uno de los sub-bloques implementados, tomando como entradas a cada sub-bloque los ficheros obtenidos de la simulación del circuito mediante la herramienta PSIM.

- **Fase de integración:** Integración de las simulaciones que realiza PSIM en el entorno analógico y Modelsim en el entorno digital.

- **Evaluación de la aplicación:** Proceso de pruebas del sistema completo hasta alcanzar una versión completamente estable.

▪ **Fase 3: Documentación**

- **Memoria del Proyecto Final de Carrera:** Redacción del presente documento de memoria del Proyecto Final de Carrera.

- **Preparación de la presentación.**

En la Figura 1.2 se muestra la estructura en árbol WBS (*"Work Breakdown Structure"*) donde se ordenan las tareas y sub-tareas arriba mencionadas con el fin de proporcionar un método sistemático.
Tal y como puede verse en dicha figura, se han definido tres familias de tareas, que conforman el primer nivel jerárquico y a partir del cual se han desarrollado las diez tareas de que se compone el proyecto.

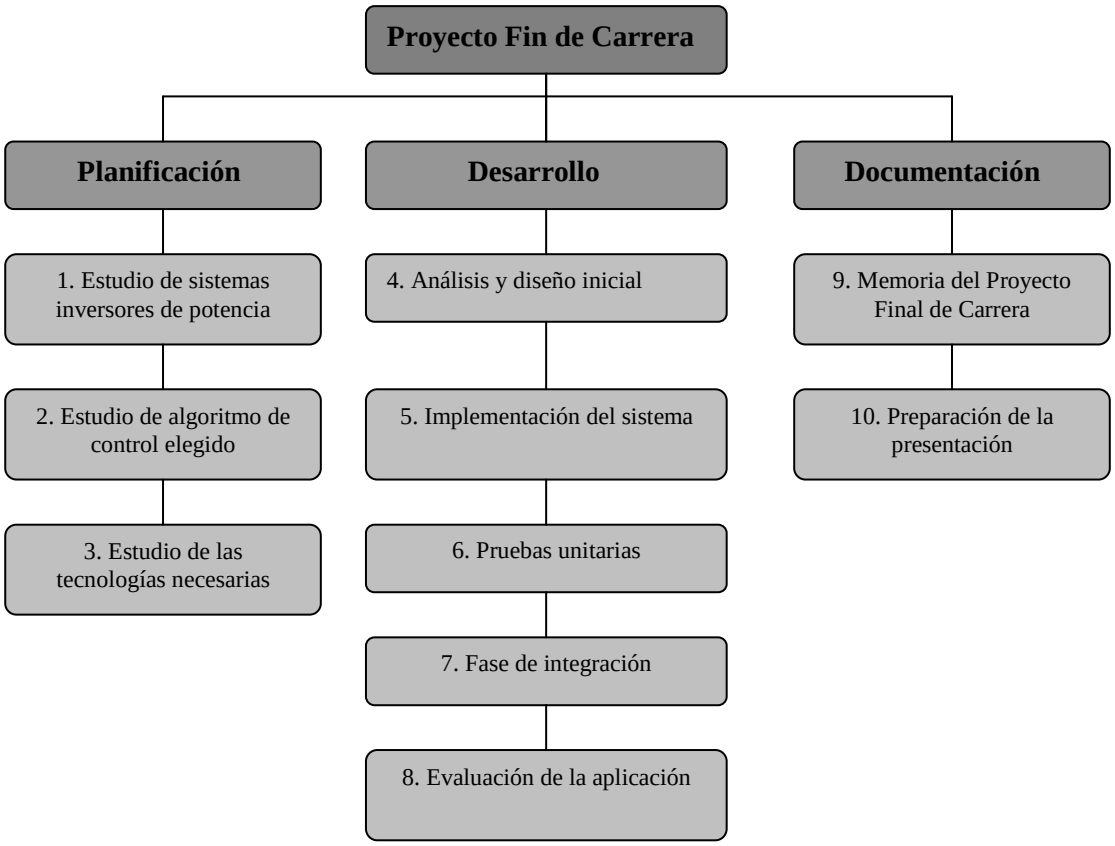


Figura 1. 3. Diagrama WBS de las tareas definidas para el proyecto.

Como puede apreciarse, únicamente el segundo y último nivel jerárquico representa tareas reales del proyecto, siendo los niveles superiores las tareas resumen.

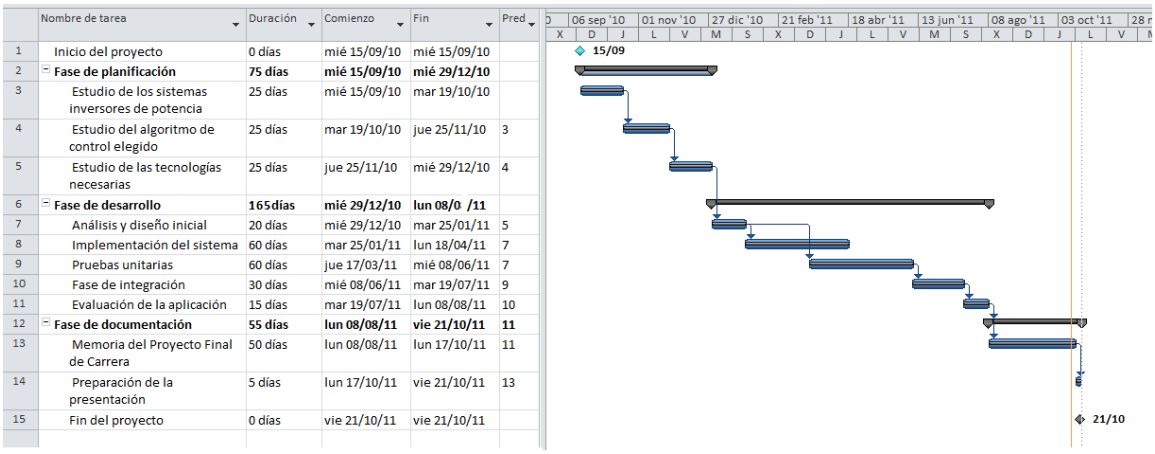
1.4 Planificación temporal

Una vez fijadas las fases y tareas en que se ha dividido el proyecto, se procede a la planificación temporal de las fases empleando para ello el *"diagrama de Gantt"*. Dicho diagrama vincula las tareas a un calendario permitiendo de esta forma realizar un seguimiento detallado del estado de avance de cada tarea.

Para la realización del *"diagrama de Gantt"*, además de incluir las diez tareas divididas en tareas resumen estudiadas en el diagrama *WBS*, se han añadido un hito de inicio de proyecto y otro de fin. Dichos hitos son tareas de duración nula empleados para marcar puntos de control de avance del proyecto.

A continuación se introducen las duraciones de cada tarea, considerando una jornada de trabajo de cuatro horas al día de lunes a viernes y de siete horas al día en sábado y domingo debido a que la realización de este Proyecto Fin de Carrera se ha llevado a cabo conjuntamente con la realización de prácticas en empresa, lo cual reduce las horas disponibles para la realización del mismo. A su vez, se han tenido en cuenta las relaciones de precedencia de cada tarea, que fijan la restricción de no empezar una determinada tarea hasta la finalización de aquella que la precede.

En la siguiente figura puede verse el *"diagrama de Gantt"* obtenido tras aplicar este procedimiento, obteniendo así una visión más global de las distintas fases que componen el proyecto, así como de su duración en el tiempo.



	Nombre de tarea	Duración	Comienzo	Fin	Pred
1	Inicio del proyecto	0 días	mié 15/09/10	mié 15/09/10	
2	**Fase de planificación**	**75 días**	**mié 15/09/10**	**mié 29/12/10**	
3	Estudio de los sistemas inversores de potencia	25 días	mié 15/09/10	mar 19/10/10	
4	Estudio del algoritmo de control elegido	25 días	mar 19/10/10	jue 25/11/10	3
5	Estudio de las tecnologías necesarias	25 días	jue 25/11/10	mié 29/12/10	4
6	**Fase de desarrollo**	**165días**	**mié 29/12/10**	**lun 08/0 /11**	
7	Análisis y diseño inicial	20 días	mié 29/12/10	mar 25/01/11	5
8	Implementación del sistema	60 días	mar 25/01/11	lun 18/04/11	7
9	Pruebas unitarias	60 días	jue 17/03/11	mié 08/06/11	7
10	Fase de integración	30 días	mié 08/06/11	mar 19/07/11	9
11	Evaluación de la aplicación	15 días	mar 19/07/11	lun 08/08/11	10
12	**Fase de documentación**	**55 días**	**lun 08/08/11**	**vie 21/10/11**	**11**
13	Memoria del Proyecto Final de Carrera	50 días	lun 08/08/11	lun 17/10/11	11
14	Preparación de la presentación	5 días	lun 17/10/11	vie 21/10/11	13
15	Fin del proyecto	0 días	vie 21/10/11	vie 21/10/11	

Figura 1.4. Diagrama de Gantt.

1.5 Medios empleados

Los medios con los que se ha contado para la realización de Proyecto Final de Carrera han sido los siguientes:

- Dispositivos Hardware:
  - Ordenador portátil.
- Aplicaciones Software:
  - Paquete Microsoft Office 2007.
  - Editor de programación Notepad ++.
  - Simulador analógico PSIM Version 9.0.3.
  - Simulador digital ModelSim SE 6.3j.
  - Software matemático MATLAB R2008a.

En el apartado Presupuesto de esta memoria, se adjunta la información referente al coste de estos medios.

Referente a la documentación examinada, se han empleado numerosos libros y artículos en el ámbito de los sistemas inversores, además de diversos manuales para las distintas

herramientas de simulación empleadas. Toda esta documentación puede encontrarse de manera detallada en el apartado de Bibliografía.

1.6 Estructura del documento

A continuación se muestra un breve resumen del contenido de cada capítulo de la memoria, con el objetivo de facilitar la lectura de este documento:

Capítulo 1: Introducción y objetivos. En este capítulo pueden encontrarse el propósito y objetivos de todo el contenido del Proyecto Final de Carrera. En este apartado se incluyen también las fases de desarrollo, la planificación temporal, los medios empleados y la estructura de la memoria.

Capítulo 2: Acrónimos. En este apartado se recopilan los términos y conceptos técnicos utilizados en la memoria, con el objetivo de facilitar su comprensión al lector.

Capítulo 3: Estado del Arte. En este capítulo se hace un estudio completo de los sistemas inversores, así como de las técnicas de control disponibles, se analiza en detalle el lenguaje de descripción hardware VHDL y se presentan las plataformas usadas para la simulación analógica y digital, PSIM y Modelsim respectivamente.

Capítulo 4: Descripción general de la aplicación. En este apartado se presenta una visión global del sistema inversor a implementar. Se explican el por qué de cada uno de sus bloques y los resultados más relevantes a tener en cuenta para una posterior implementación mediante VHDL.

Capítulo 5: Diseño digital del control del inversor. En esta sección se realiza una descripción detallada de cada uno de los módulos que se han ido desarrollando mediante VHDL, previos a la integración final del sistema. Para ello se muestran las rutas de datos de cada uno de los bloques de que consta la descripción hardware desarrollada, así como la comprobación del correcto funcionamiento de cada uno de los bloques mediante la elaboración de Test bench.

Capítulo 6: Resultados de simulación. Donde se expone el sistema final implementado a integrar en la herramienta de simulación PSIM.

Capítulo 7: Conclusiones y trabajo futuro. Finalmente se exponen las principales ideas, cuestiones y conclusiones derivadas de la realización del proyecto, así como las posibles líneas de investigación que quedan abiertas a partir de este proyecto.

Presupuesto. Esta sección contiene un análisis de los costes del diseño y desarrollo del proyecto, detallando el coste de personal y del material necesario para llevar a cabo su realización.

Bibliografía. Donde pueden encontrarse las citas bibliográficas consultadas para la realización del proyecto y de la memoria.

Anexos. En este apartado se muestra el código VHDL de la aplicación final.

Capítulo 2

Acrónimos

PWM:	*Pulse Width Modulation.*
ASIC:	*Application Specific Integrated Circuit.*
CPLD:	*Complex Programmable Logic Device.*
FPGA:	*Field Programmable Logic Device.*
WBS:	*Work-breakdown structure.*
CAD:	*Computer-Aided Design.*
IEEE:	*Institute of Electrical and Electronics Engineers.*
VHSIC:	*Very High Speed Integrated Circuit.*
VHDL:	*VHSIC Hardware Description Language.*
PLL:	*Phase Locked Loop.*
RMS:	*Root Mean Square*
MSPWM:	*Modified Sinusoidal Pulse Width Modulation.*
DSP:	*Digital Signal Processor.*
PI:	*Proportional Integral.*
A/D:	*Analogic-digital.*
AMS:	*Analog and Mixed Signal.*
MOS:	*Metal Oxide Semiconductor.*
NMOS:	*Negative Channel Metal Oxide Semiconductor.*
MSI:	*Medium Scale Integration.*
LSI:	*Low Scale Integration.*
IDL:	*Interface Description Language.*
RTL:	*Resistor Transfer Logic.*
DLL:	*Dynamic Linking Library.*
LPF:	*Low Pass Filter.*

Capítulo 3

Estado del arte

En este capítulo se analiza el contexto en que se enmarca el Proyecto Fin de Carrera.
En primer lugar, se presentan los sistemas inversores, explicando los módulos que componen su arquitectura, examinando los distintos criterios a tener en cuenta para su diseño y analizando las distintas técnicas existentes en el control digital para sistemas inversores.
Posteriormente se realizará un estudio completo del lenguaje de descripción de hardware elegido VHDL, así como de los entornos de simulación PSIM y Modelsim que serán empleados para la validación del control digital diseñado en este proyecto.

3.1 Sistemas inversores

3.1.1 Introducción

Desde el punto de vista de calidad de la señal, los dos puntos principales a tener en cuenta en un inversor son el factor de potencia y la distorsión armónica. Generalmente, los inversores operan con factores de potencia bastante inferiores a la unidad, lo que en las conexiones a red debe evitarse, ya que con factores de potencia bajos el inversor

demandará energía reactiva a la red, afectando esto a la tensión del sistema, pudiendo degradar la calidad del servicio de otros posibles consumidores conectados a ella.

La eficiencia global de un sistema inversor conectado a la red depende en gran parte de la topología y el control del inversor.

A continuación se presentan las distintas topologías de inversores conectados a la red, así como las técnicas de control más usadas, tales como el control lineal de la corriente, el control por histéresis de la corriente o el control predictivo de la corriente.

3.1.2 Topologías de inversores de potencia

Los inversores no son más que convertidores estáticos de energía que convierten la corriente continua CC en corriente alterna CA, con la posibilidad de alimentar una carga en alterna, regulando la tensión, la frecuencia o bien ambas.
Más concretamente, los inversores transfieren potencia desde una fuente de continua a una carga de alterna.

Varias son las aplicaciones típicas que pueden tener los inversores de potencia:

- Accionamientos de motores de CA de velocidad ajustable.
- Sistemas de alimentación ininterrumpida (SAI).
- Dispositivos de corriente alterna que funcionan a partir de una batería.
- Hornos de inducción.

Inversores monofásicos

Suelen distinguirse tres configuraciones de inversores que corresponden a las tres maneras más razonables de realizar la función de inversión de tensión o corriente suministrada por la fuente de CC con los medios disponibles **[7]**.

- ***Push-pull*:** Debe tenerse en cuenta la relación de espiras entre cada uno de los primarios (considerando que está en medio puente) y el secundario.

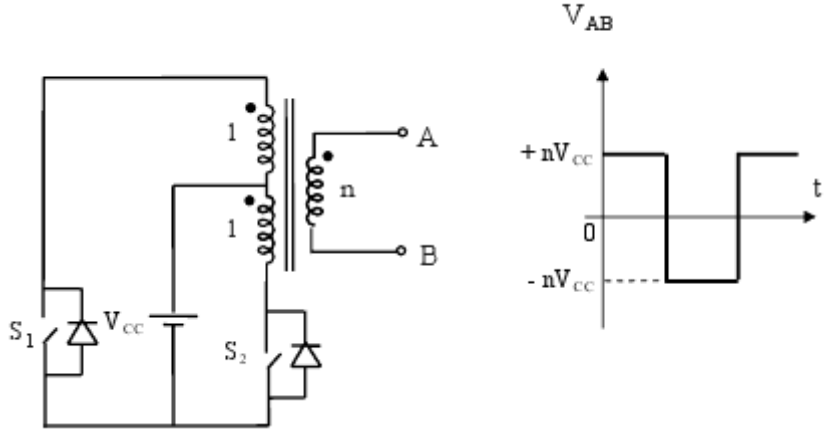


Figura 3 1. Inversor "Push-pull".

- **Medio puente:** Esta topología puede implementarse con una batería y dos condensadores en medio puente o bien con una batería en medio puente.

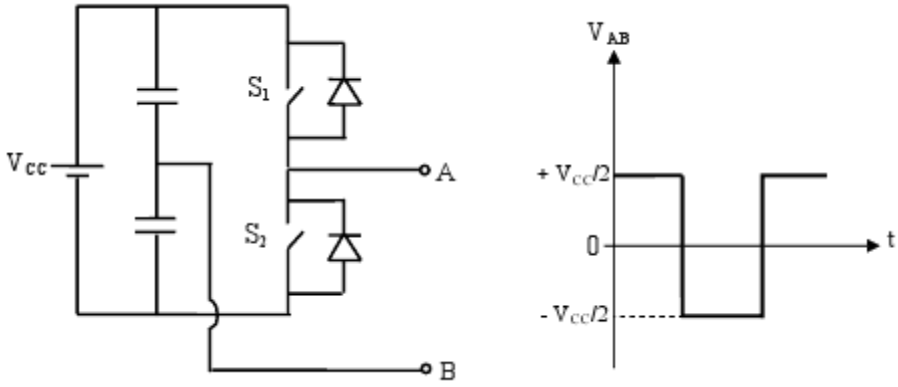


Figura 3 2. Inversor en medio puente.

- **Puente completo:** Está formado por cuatro interruptores de potencia totalmente controlados, típicamente transistores MOSFET o IGBTs, tal y como se muestra en la siguiente figura.

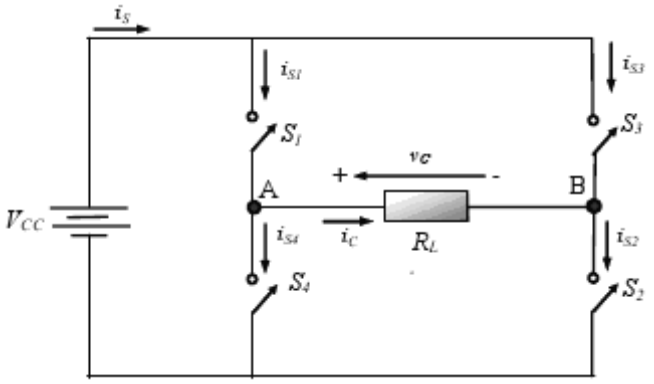


Figura 3 3. Inversor en puente completo.

Del análisis del inversor en puente completo se puede deducir el funcionamiento de los dos anteriores (*"push-pull"* y medio puente).

La tensión de salida v_c puede ser $+V_{cc}$, $-V_{cc}$, ó 0, según el estado de los interruptores **[8]**. La siguiente tabla muestra la tensión de salida obtenida al cerrar cada una de las parejas de interruptores.

INTERRUPTORES CERRADOS	**TENSIÓN DE SALIDA Vc**
S_1 y S_2	+Vcc
S_3 y S_4	-Vcc
S_1 y S_3	0
S_2 y S_4	0

Tabla 3.1.Tensión de salida según estado de los interruptores

Como puede observarse, las parejas de interruptores S_1, S_4 y S_2, S_3 no deberían estar cerrados al mismo tiempo, para evitar un cortocircuito en la fuente de continua.
Puesto que los interruptores reales no se abren y se cierran instantáneamente, deberán tenerse en cuenta los tiempos de conmutación al diseñar el control de los interruptores.

El tiempo permitido para la conmutación se denomina tiempo muerto (*"blanking time"*). Para obtener una tensión de salida v_C igual a cero se pueden cerrar al mismo tiempo los interruptores S_1 y S_3 o bien S_2 y S_4.
Otra forma de obtener una tensión nula a la salida sería eliminando las señales de control en los interruptores, es decir, manteniendo abiertos todos los interruptores.

Inversores trifásicos

Los inversores trifásicos se emplean en la alimentación de cargas trifásicas. De este modo, aplicaciones como fuentes ininterrumpidas de tensión alterna trifásica, accionamientos de motores de corriente alterna trifásicos y conexión de fuentes que producen energía en continua con las cargas trifásicas, usan este tipo de inversores **[9]**.

En la Figura 3.4 se muestra el inversor trifásico empleado con mayor frecuencia.

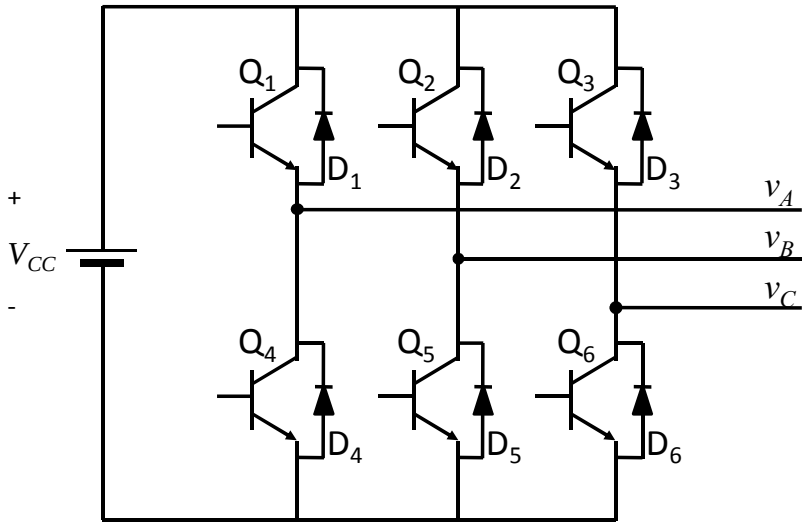


Figura 3 4. Inversor trifásico de dos niveles.

Este convertidor se conoce como inversor trifásico en dos niveles puesto que cada uno de sus terminales del lado AC puede alcanzar los dos extremos de tensión V_{CC} y $-V_{CC}$.
Los lados de continua de los convertidores en medio puente están conectados en paralelo con una fuente de continua común, mientras que el lado de alterna de cada uno de los convertidores está relacionado con una fase de un sistema trifásico.
De aquí en adelante se indexará cada uno de los convertidores en medio puente como *a, b* y *c* , que estarán asociadas a las correspondientes fases del sistema trifásico.

En electrónica de potencia los componentes electrónicos suelen funcionar en conmutación para evitar el calentamiento excesivo del silicio, por lo que siempre se encontrarán en estado de conducción o de bloqueo.

De esta forma, la carga conectada a la salida del inversor recibe una tensión en forma de pulsos. Controlando los instantes de entrada en conducción y apertura de los interruptores es posible reproducir en la carga una forma de onda determinada, como por ejemplo una tensión senoidal.

En el inversor trifásico propuesto hay seis células de conmutación (seis interruptores de potencia: Q_1 a Q_6), y seis diodos de libre circulación (D_1 a D_6), colocados en antiparalelo

con los interruptores y que se encargan de garantizar la continuidad de la corriente en la carga además de permitir la reversibilidad de la potencia al permitir inyectar corriente desde la carga a la batería de continua.

Cada una de las ramas del inversor está formada por dos interruptores en paralelo con los diodos de libre circulación, estando la salida de cada fase en el punto medio de la rama.

Las señales de control de los interruptores de cada rama deben ser complementarias para evitar que se cortocircuite la fuente V_{CC}. Por otro lado, como ya se ha mencionado a hablar de sistemas monofásicos, es necesario tener en cuenta que los interruptores necesitan un tiempo denominado *tiempo muerto*, tanto para realizar la apertura como para el cierre, por lo que, antes de cerrar un interruptor para permitir el paso de corriente, es necesario esperar este *tiempo muerto* para que el interruptor complementario haya tenido tiempo suficiente para abrirse y no se produzca un cortocircuito en la fuente.

En la siguiente figura se muestran las diferentes combinaciones posibles, donde puede apreciarse que en el primer estado y en el último las tensiones son nulas, por lo que estos estados se conocen como *estados de libre circulación*.

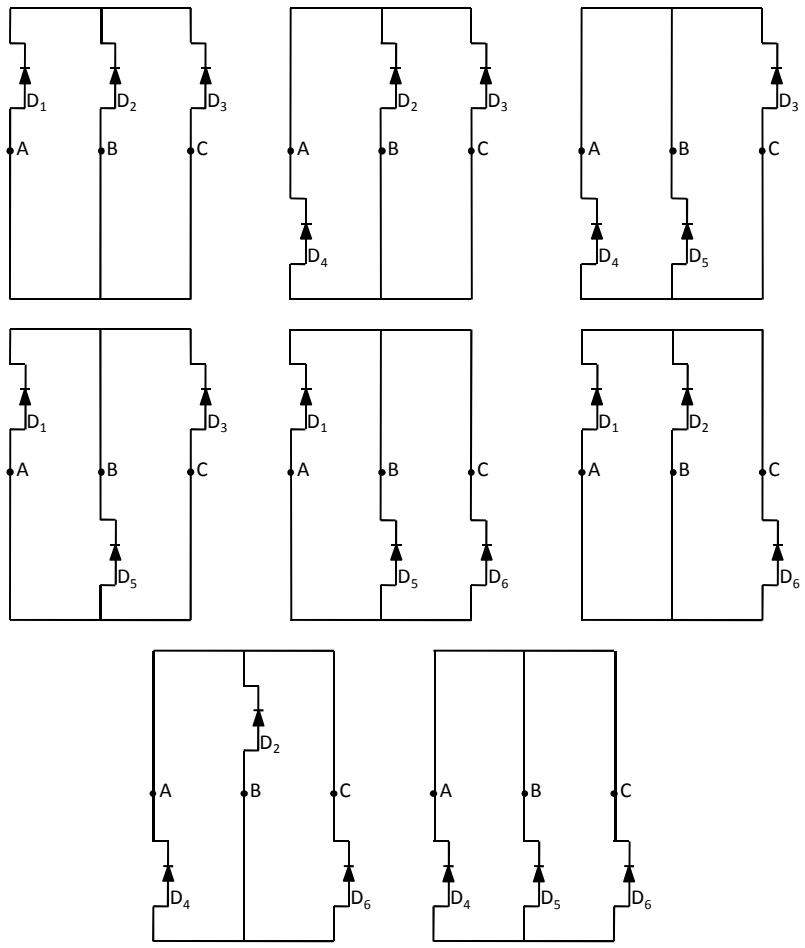


Figura 3 5. Configuraciones del inversor en función del estado de los interruptores.

3.1.3 Técnicas de modulación

A continuación se presentan las técnicas de modulación del inversor en puente completo **[10]**:

- **Inversor con modulación por onda cuadrada.**
  La técnica de modulación o el esquema de conmutación más sencillo del inversor en puente completo es el que genera una tensión de salida en forma de onda cuadrada. En este caso los interruptores conectan la carga a $+V_{CC}$ cuando S_1 y S_2 están cerrados (estando S_3 y S_4 abiertos) y a $-V_{CC}$ cuando S_3 y S_4 están cerrados (estando S_1 y S_2 abiertos). La conmutación periódica de la tensión de la carga entre $+V_{CC}$ y $-V_{CC}$ genera en la carga una tensión con forma de onda cuadrada. Aunque esta salida alterna no es senoidal pura, puede ser una onda de alterna adecuada para algunas aplicaciones.

  La forma de onda de la corriente en la carga depende de los componentes de la carga.

  En una carga resistiva, la forma de onda de la corriente se corresponde con la forma de la tensión de salida. Una carga inductiva tendrá una corriente más senoidal que la tensión, a causa de las propiedades de filtrado de las inductancias.

  Una carga inductiva requiere ciertas consideraciones a la hora de diseñar los interruptores del inversor, ya que las corrientes de los interruptores deben ser bidireccionales. Para ello, se suelen poner diodos en antiparalelo con cada uno de los interruptores. En el caso del inversor en puente se utilizarían cuatro diodos en antiparalelo con cada uno de los interruptores. Para el caso del medio puente y del *"push-pull"* se utilizarían dos diodos, uno para cada interruptor.

  La siguiente figura muestra la forma de onda de la tensión de salida v_C para un inversor en puente de onda completa con modulación por onda cuadrada. Este tipo de modulación no permite el control de la amplitud ni del valor eficaz de la tensión de salida, la cual podría variarse solamente si la tensión de entrada V_{CC} fuese ajustable.

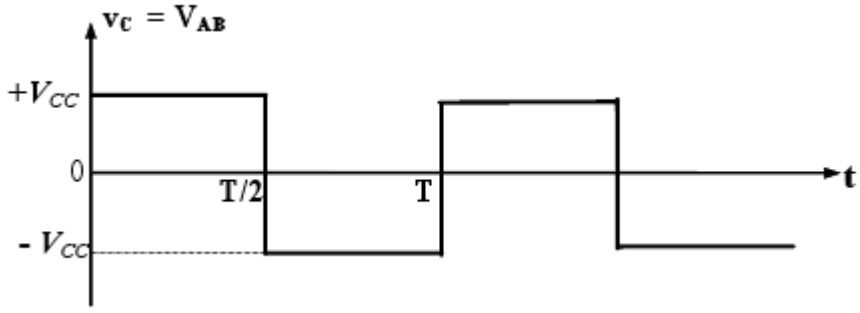


Figura 3 6. Formas de onda de tensión en la carga RL del inversor en puente controlado por onda cuadrada.

La transformada de *Fourier* de una onda cuadrada es conocida y se muestra a continuación. Como se puede observar, presenta todos los armónicos impares, con una disminución de amplitud proporcional a la frecuencia de los mismos.

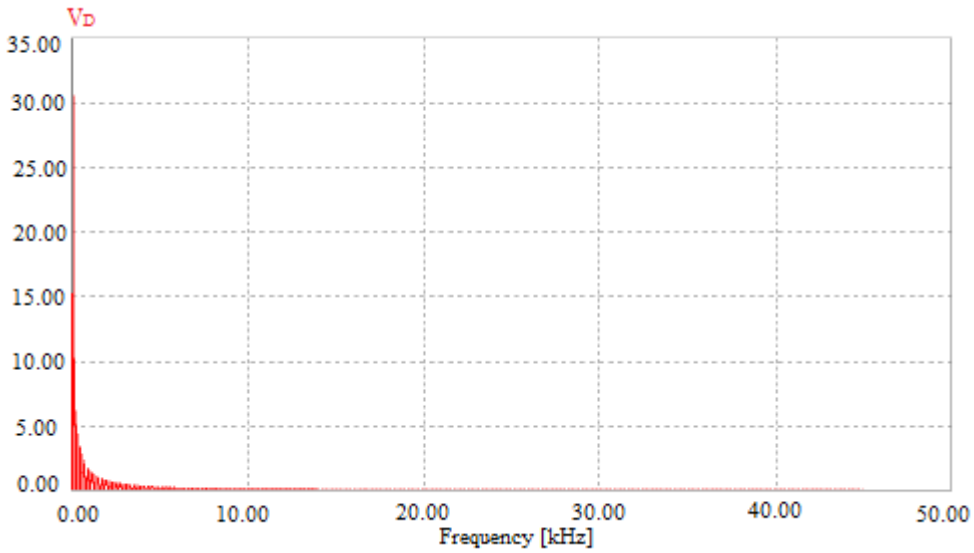


Figura 3 7. Espectro de la tensión de salida de un inversor por onda cuadrada.

- **Inversor con modulación por onda cuasi-cuadrada.**

  De las figuras previas se deduce que aunque a la salida se ha obtenido una tensión alterna, esta no se parece en absoluto a una sinusoide pura. De hecho, una onda cuadrada periódica pura tiene infinitos armónicos sobre la frecuencia fundamental.

  Para solucionar este inconvenientes existen varias alternativas. La primera es añadir un filtro tipo *LC* a la salida, lo cual es costoso dado el elevado número de armónicos de baja frecuencia que se deben filtrar. La segunda alternativa es mejorar el control de los interruptores de potencia.

  Una alternativa que permite ajustar el valor eficaz de la tensión de salida y eliminar los armónicos de baja frecuencia es la llamada onda cuasi-cuadrada o cancelación de tensión, en la que se mantiene un nivel de tensión nulo sobre la carga durante parte del período, mejorando así el contenido de armónicos de la tensión de salida.

  La siguiente figura se muestra la tensión obtenida utilizando la modulación por onda cuasi-cuadrada.

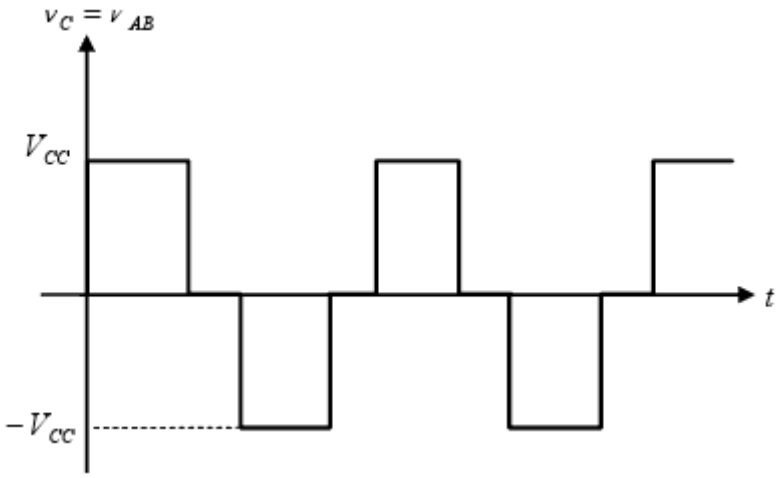


Figura 3 8. Formas de onda de tensión en la carga R_L del inversor en puente completo controlado por cancelación de tensión (modulación por onda casi-cuadrada).

Para obtener este tipo de onda, una posibilidad sería la siguiente: cuando se desea tensión positiva en la carga se mantienen S_1 y S_2 conduciendo (S_3 y S_4 abiertos). La tensión negativa se obtiene de forma complementaria (S_3 y S_4 cerrados y S_1 y S_2 abiertos). Los intervalos de tensión nula se obtendrán, como ya se ha comentado, cerrando simultáneamente los interruptores S_1 y S_3 manteniendo S_2 y S_4 abiertos o bien cerrando S_2 y S_4 mientras S_1 y S_3 se mantienen abiertos. Otra forma de obtener tensión nula a la salida es manteniendo todos los interruptores abiertos durante el intervalo de tiempo deseado.

Si se efectúa un análisis de Fourier de la forma de onda cuasi-cuadrada, se observaría que están presentes los armónicos impares de la frecuencia de conmutación a baja frecuencia.

- **Control por modulación de anchura de pulsos PWM.**

Si se quiere mejorar aún más el contenido de armónicos en la salida de un inversor, es necesario utilizar lo que se conoce como modulación de anchura de pulsos PWM (“*Pulse Width Modulation”*).

La idea básica es comparar una tensión de referencia senoidal de baja frecuencia (que sea imagen de la tensión de salida buscada) con una señal triangular simétrica de alta frecuencia cuya frecuencia determine la frecuencia de conmutación. La frecuencia de dicha onda triangular (denominada portadora) debe ser, como mínimo veinte veces superior a la máxima frecuencia de la onda de referencia, para que se obtenga una reproducción aceptable de la forma de onda sobre una carga, una vez realizado el filtrado. La señal resultante de dicha comparación generará la lógica para abrir y cerrar los semiconductores de potencia.

La siguiente figura muestra la modulación de una onda senoidal, produciendo en la salida una tensión con dos niveles, cuya frecuencia es la de la onda triangular.

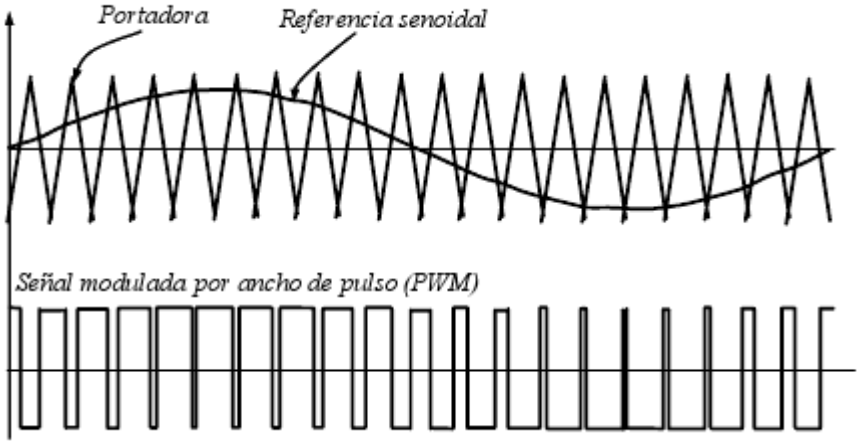


Figura 3 9. Generación de una señal PWM.

A partir de la señal PWM se generan los pulsos de apertura y cierre de los interruptores. De esta forma, si la señal PWM tiene un valor alto, se cierran los interruptores S_1 y S_2. En caso contrario se cierran los interruptores *S3* y *S4*.
Por tanto, la tensión de salida, que es aplicada a la carga, está formada por una sucesión de ondas rectangulares de amplitud igual a la tensión de alimentación en continua y duración variable. El contenido de armónicos de la tensión de salida se desplaza hacia las frecuencias elevadas y es más fácil de filtrar.
La Figura 3.10 muestra la señal PWM en un cuarto de la senoide completa para una apreciación más detallada.

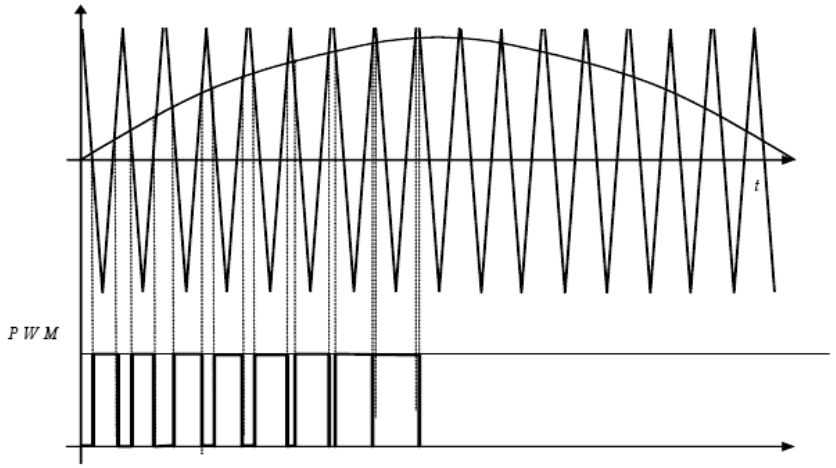


Figura 3 10. Generación de una señal PWM en un cuarto de senoide completa.

Por lo tanto, un filtro paso bajo con frecuencia de corte por encima de la frecuencia de referencia será perfectamente capaz de producir una atenuación bastante efectiva.

3.1.4 Técnicas en la modulación PWM

Las técnicas más comúnmente usadas en la modulación PWM son la modulación de un solo ancho de pulso, modulación de varios anchos de pulso, modulación senoidal del ancho de pulso, modulación senoidal modificada del ancho de pulso y control por desplazamiento de fase. A continuación, en este apartado se presenta una breve introducción de cada una de estas técnicas **[11]**:

- **Modulación de un solo ancho de pulso.**
  En el control por esta técnica existe un solo pulso por cada medio ciclo. El ancho del pulso se hace variar, a fin de controlar el voltaje de salida del inversor.

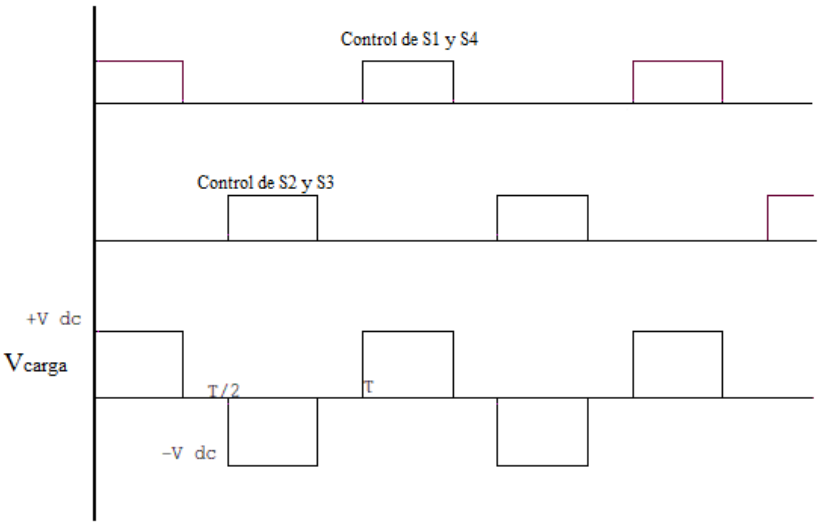


Figura 3 11. Modulación de un solo ancho de pulso.

- **Modulación de varios anchos de pulso.**
  También conocido como modulación uniforme de ancho de pulso (UPWM).
  Utilizando varios pulsos en cada semiciclo de voltaje de salida puede reducirse el contenido armónico.
  Mediante la comparación de una señal de referencia con una señal portadora, se generan los pulsos de disparo.
  La frecuencia de la señal de referencia establece la frecuencia de la señal de salida, f_0, y la frecuencia de la portadora, f_C, determina el número de pulsos por cada ciclo.

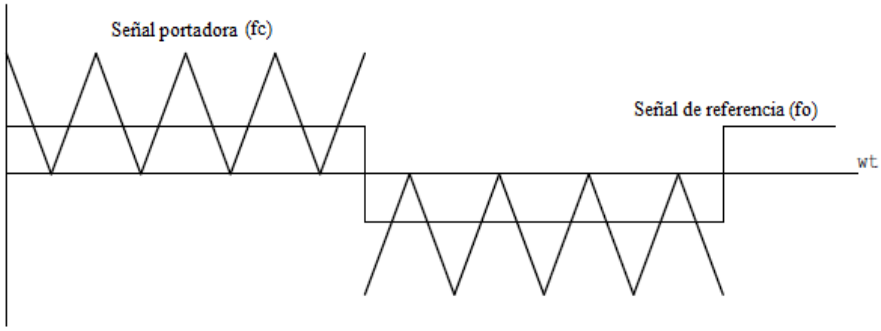


Figura 3 12. Generación de la señal de excitación.

El índice de modulación controla el voltaje de salida:

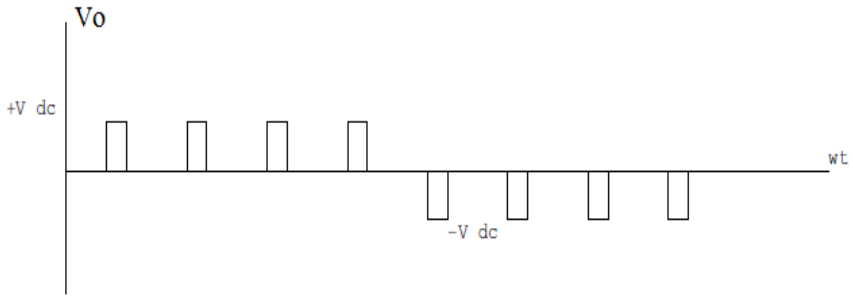


Figura 3 13. Voltaje de salida.

- **Modulación senoidal del ancho de pulso.**
  En este caso, en lugar de mantener igual el ancho de todos los pulsos, en la modulación senoidal se varía en proporción con la amplitud de una onda senoidal evaluada en el centro del mismo pulso.
  Las señales de compuerta se generan al comparar una señal senoidal de referencia con una onda portadora triangular. La frecuencia de la señal de referencia, *fr*, determina la frecuencia de salida del inversor, *fo*, y su amplitud de pico controla el índice de modulación, *M*, y en consecuencia, el voltaje RMS (*"Root Mean Square"*) de salida.

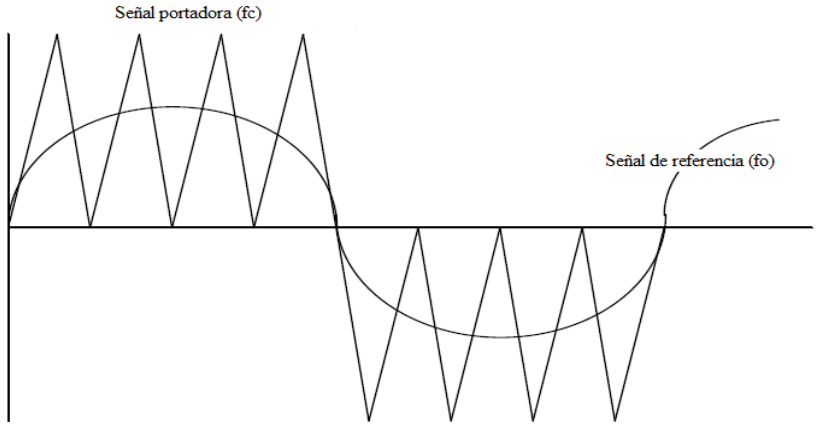


Figura 3 14. Voltaje de salida.

- **Modulación senoidal modificada del ancho de pulso.**
  En la modulación senoidal de ancho de pulso, los anchos de los pulsos más cercanos al pico de la onda senoidal no cambian considerablemente, luego en la modulación senoidal modificada (MSPWM) durante los sesenta grados alrededor del pico (treinta antes y treinta después) los interruptores están encendidos. Este tipo de modulación permite que se reduzca el número de conmutaciones de los dispositivos de potencia y las pérdidas de conmutación, incrementando la componente fundamental y mejorando las características armónicas

- **Control por desplazamiento de fase.**

  Para el caso particular de los inversores en puente completo, existe la posibilidad de emplear un esquema de control de los interruptores en el cual estos trabajan a baja frecuencia, entendiendo por baja frecuencia la de la tensión de salida que se desea obtener. En este caso, todos los interruptores trabajan con un ciclo de trabajo del 0.5, de forma similar al esquema de onda cuadrada, pero con la diferencia de que ambas ramas se controlan de forma independiente, y dentro de un mismo ciclo, se permite que los interruptores superiores o los inferiores estén en saturación, de forma que la tensión de salida en el conjunto filtro+carga es nula en determinados instantes de tiempo.
  En este caso es posible controlar la tensión de salida, en cuanto a amplitud y frecuencia.

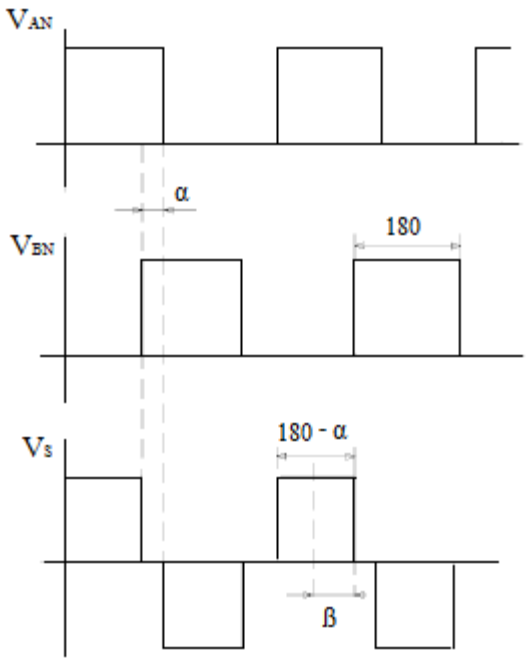


Figura 3 15. Control por desplazamiento de fase.

Por lo general, en un sistema inversor, interesa controlar la corriente que dicho sistema inyecta a la red, por lo que esta es la variable que será necesario realimentar en el inversor al cerrar el lazo de regulación.

A continuación, en el siguiente apartado, se muestran las distintas estructuras de control empleadas hoy día y las soluciones que aportan para inversores trifásicos.

3.1.5 Estructuras de control

En la mayoría de aplicaciones de inversores con modulación de ancho de pulso PWM, los convertidores poseen una estructura de control compuesta de un lazo interno de control de corriente.

La corriente de salida del inversor deberá ser lo más sinusoidal posible y con baja distorsión armónica. Consecuentemente, el rendimiento del inversor dependerá en gran parte de la estrategia de control que se aplique, convirtiéndose de este modo el controlador de corriente en uno de los más importantes en las aplicaciones de inversores.

Algunas de las ventajas que aporta son: un control de la corriente instantánea y alta precisión, muy buena dinámica, máxima protección de la corriente y compensación de la caída de tensión de los semiconductores y de los tiempos muertos del convertidor.

Las técnicas de control de la corriente en inversores se clasifican generalmente en controladores lineales y no lineales.

Los controladores lineales funcionan con el modulador de tensión PWM tradicional. Este concepto permite aprovechar las ventajas del lazo abierto del modulador (PWM sinusoidal):

- Frecuencia de conmutación constante o limitada, para garantizar la operación fiable de los dispositivos del convertidor.
- Espectro armónico bien definido.
- Patrón de conmutación óptimo.

Los avances en el campo de los procesadores digitales (DSP (*"Digital Signal Processor"*), FPGA) han dado lugar a la aparición de diferentes técnicas en el control de la corriente del convertidor estático.

A continuación se detallan las técnicas de control de corriente que han mostrado mayor efectividad en aplicaciones de inversores conectados a la red **[12]**.

- **Control lineal de corriente.**
  Utiliza la modulación PWM y la señal moduladora que se compara con la portadora triangular proviene de la salida de un regulador lineal, generalmente un regulador proporcional-integral PI.

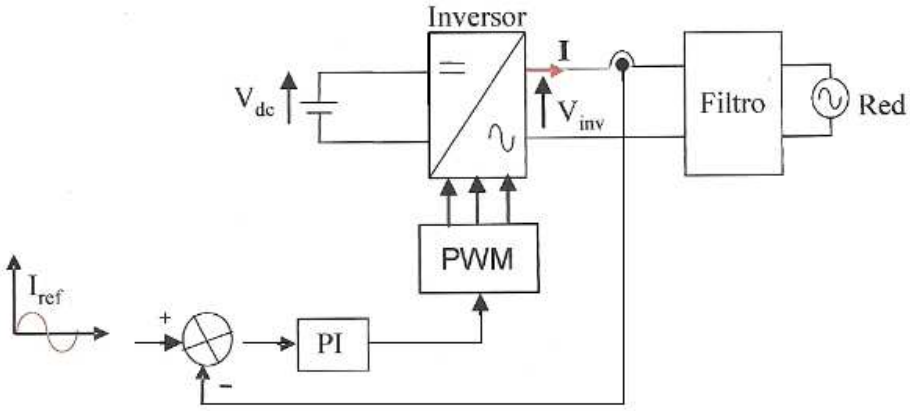


Figura 3 16. Control lineal de corriente.

Aunque para sistemas trifásicos el control lineal se puede implementar sin problemas sobre un sistema de referencia estacionario, una variante consiste en la utilización de un sistema de referencia rotativo, tal como se verá en la sección 3.1.6.

La elección de los parámetros del regulador es directa cuando se trabaja en tiempo continuo y el control puede llevarse a cabo mediante circuitos analógicos o digitales.

El control lineal proporciona inmunidad al ruido, teniendo en cuenta que el ancho de banda limitado del regulador elimina las componentes de alta frecuencia de la señal de error de corriente.
Dicha limitación del ancho de banda se debe a la restricción impuesta por la máxima pendiente de la señal moduladora, y en ningún momento se puede superar la pendiente de la portadora.

- **Controlador no lineal de corriente.**
  El control no lineal de corriente incluye por lo general, el control por histéresis **[13]**.

  El control por histéresis se utiliza en la regulación de la corriente de convertidores. La corriente inyectada se compara con la corriente de referencia, y el error resultante se aplica a un comparador de histéresis, el cual presenta una característica entrada-salida que depende del sentido de variación de la señal de entrada.
  Tal como puede verse en la Figura 3.17, cuando el valor de entrada al comparador de histéresis comienza a crecer desde un valor igual a cero, la salida se mantiene a un nivel alto hasta que la entrada alcanza un valor *"High"*. Si se recorre el camino en sentido contrario, es decir, la entrada va disminuyendo desde un valor alto, la salida se mantiene a un nivel bajo mientras la entrada sea superior al valor "Low".

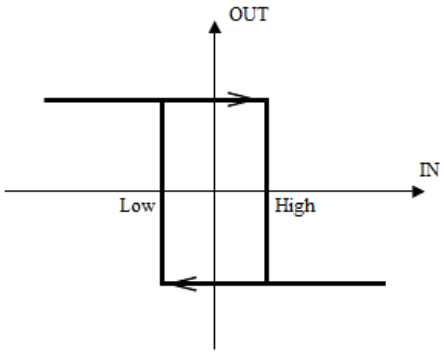


Figura 3 17. Relación entrada-salida de un comparador de histéresis.

Las señales de conmutación obtenidas son fijadas para mantener el error en la corriente inyectada dentro de la banda de histéresis. Debido a la ausencia de retardos y a la inherente no linealidad, este control permite proporcionar la respuesta dinámica más rápida.

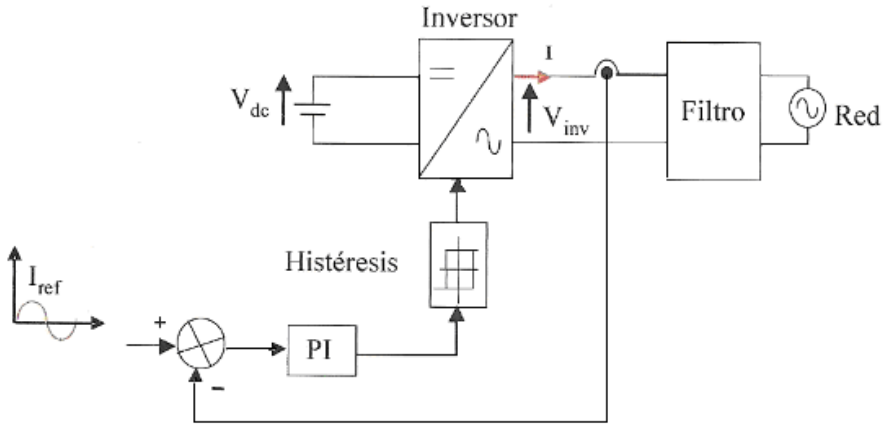


Figura 3 18. Control por histéresis.

El control por histéresis es robusto, fácil de implementar y proporciona una limitación instantánea de la corriente. No obstante, a pesar de todas estas ventajas, esta técnica de control también presenta algún inconveniente. La principal desventaja es que da lugar a una frecuencia de modulación variable en el inversor de potencia, lo que complica el diseño del filtro pasivo de salida del mismo, y puede resultar en resonancias con la red.

El control por histéresis se aplica de forma sencilla a inversores en los que las ramas de transistores trabajan de manera independiente, no obstante, en inversores en puente completo, la interacción entre las corrientes de las fases afecta de manera negativa en el funcionamiento de este controlador.

- **Control predictivo de corriente.**

Este tipo de control es capaz de predecir el valor que debería adoptar la tensión de salida del inversor para asegurar que la corriente inyectada alcance

el valor de referencia, en cada periodo de modulación, y en base al error actual y a los parámetros del sistema.

Cuando la tensión de salida del inversor se elige de forma que el error de corriente es eliminado al final del siguiente periodo de conmutación, este control se conoce como *"dead-beat"* **[14].**

Esta técnica de control asegura que en régimen permanente, la corriente siga a la referencia con un retraso de dos periodos de muestreo. Este sistema de control es susceptible a inestabilidades y oscilaciones.

El control "*dead-beat"* suele programarse en un procesador digital de señal, y generalmente emplea modulación vectorial en el convertidor, adecuado también para implementación digital.

Esta técnica de control requiere una elevada potencia de procesado, y necesita una frecuencia de muestreo relativamente elevada.

3.1.6 Estructura de control con sistema de referencia síncrono

El control de un sistema de referencia síncrono o control dq, utiliza la transformación abc-dq para la transformación de la corriente de la red y las tensiones en un sistema de referencias que giran con la tensión de la red. De esta forma, las variables de control se transforman en valores continuos y el filtrado y control pueden lograrse con mayor facilidad. El esquema del control dq se muestra en la siguiente figura.

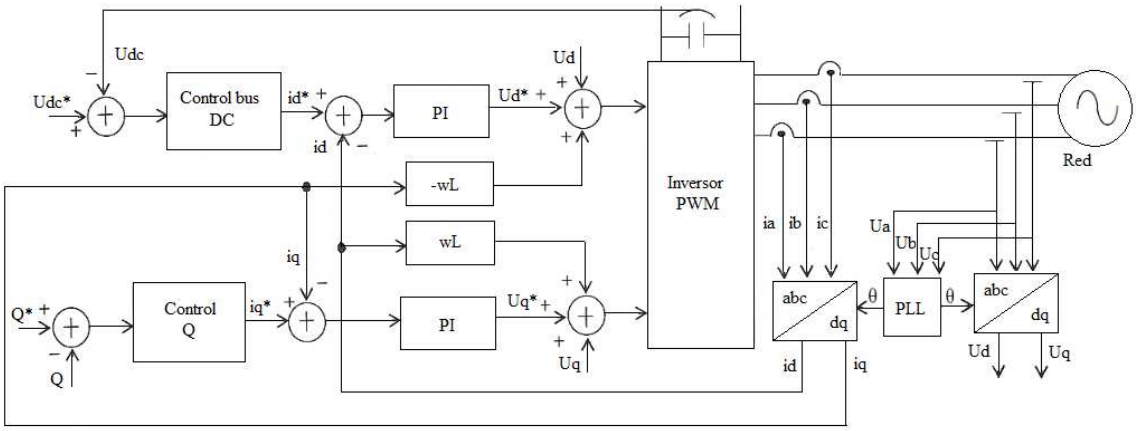


Figura 3 19. Estructura de control de un sistema de referencia síncrono.

En esta estructura, la tensión de continua se controla con relación a la necesidad de potencia de salida, mediante la potencia de salida se determina la referencia para la corriente activa y sin embargo, la referencia para la reactiva suele ser cero, si se trata solamente de inyectar potencia activa.

Cuando la potencia reactiva debe controlarse, la referencia de dicha potencia reactiva debe imponerse al sistema.

La estructura de control dq es normalmente asociada con controladores proporcionales integrales (PI) ya que los controladores tienen un buen comportamiento para la regulación de las variables de continua.

La matriz de la función de transferencia del controlador en coordenadas dq puede escribirse como:

$$G_{pi}^{(dq)}(s) = \begin{bmatrix} K_p + \frac{K_i}{s} & 0 \\ 0 & K_p + \frac{K_i}{s} \end{bmatrix} \quad (3.1)$$

Donde K_p, es la ganancia proporcional y K_i es la ganancia integral del controlador.

Dado que la corriente controlada tiene que estar en fase con la tensión de red, el ángulo de fase utilizado por el módulo de la transformación abc-dq tendrá que ser extraído de las tensiones de red.

Una posible solución es el filtrado de las tensiones de la red y la utilización de la función arcotangente para extraer el ángulo de fase. La técnica PLL (“*phase locked loop*”) se utiliza siempre para extraer el ángulo de fase de las tensiones de red en el caso de los sistemas de generación distribuida.

Con el fin de mejorar el rendimiento del controlador PI, siempre se utiliza el “*feed-forward*” para lograr mejor respuesta dinámica y mayor estabilidad del sistema respecto al retraso del lazo introducido en el sistema.
Para resolver este problema, un método avanzado de filtrado de la tensión “*feed-forward*” de la red tiene que ser considerado. La utilización del “*feed-forward*” siempre exige un sistema de transformaciones, lo cual complica su implementación.

No obstante, con todas estas mejoras, la capacidad de compensación del armónico de bajo orden en el caso de los reguladores PI es pobre.

3.1.7 Estructura de control con sistema de referencia estacionario

En este caso, las corrientes de la red están transformadas a un sistema de referencia estacionario utilizando la transformación *abc-αβ*. Puesto que las variables de control son sinusoidales, bajo estas circunstancias y debido al inconveniente del controlador PI en su defecto para eliminar el error en el régimen permanente cuando se controlan señales sinusoidales, es necesaria la utilización de otro tipo de controlador.

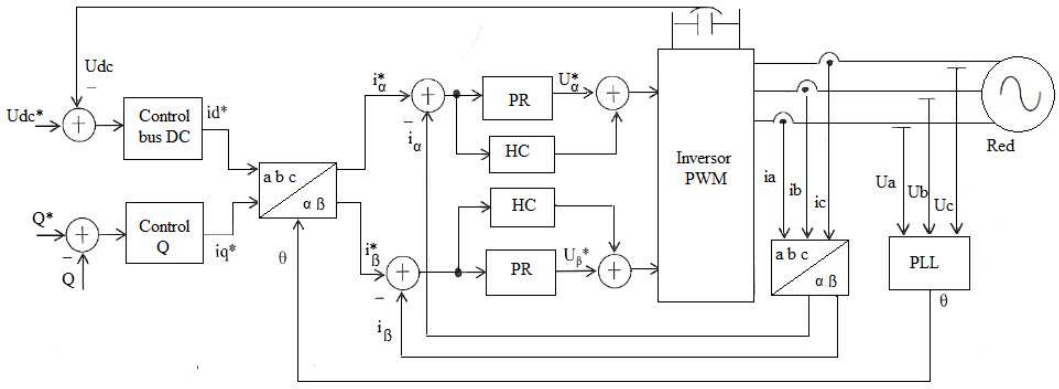


Figura 3 20. Estructura de control de un sistema de control estacionario.

El Controlador Proporcional Resonante, PR, es una solución alternativa ante el bajo rendimiento del controlador proporcional integral PI **[12]**.
En el caso de PR, el controlador de la matriz en un sistema de referencia estacionario viene dado por:

$$G_{pi}^{(dq)}(s) = \begin{bmatrix} K_p + \dfrac{K_i s}{s^2 + w^2} & 0 \\ 0 & K_p + \dfrac{K_i s}{s^2 + w^2} \end{bmatrix} \qquad (3.2)$$

Donde ω es la frecuencia de resonancia del controlador, K_p es la ganancia proporcional, y K_i es la ganancia integral del controlador.

El funcionamiento básico del regulador PR consiste en la introducción de una ganancia infinita a la frecuencia de resonancia para eliminar el error en régimen permanente a esta frecuencia entre la señal del controlador y su referencia, y no exige la utilización de *"feed forward"*.

3.1.8 Sistema de control *abc*

El objetivo es tener un controlador individual para cada corriente de la red, utilizando por lo general, el control por histéresis o el *"dead-beat control"*.

El rendimiento de estos controladores es proporcional a la frecuencia de muestreo, por lo que un rápido desarrollo de los sistemas digitales, tales como procesadores de señal digital o FPGA supone una gran ventaja para este tipo de implementación.
El controlador de corriente puede ser un proporcional integral, PI, o un proporcional resonante, PR.

En la siguiente figura se presenta una posible implementación del contol abc, donde la salida del regulador de la tensión de continua establece la referencia de la corriente.

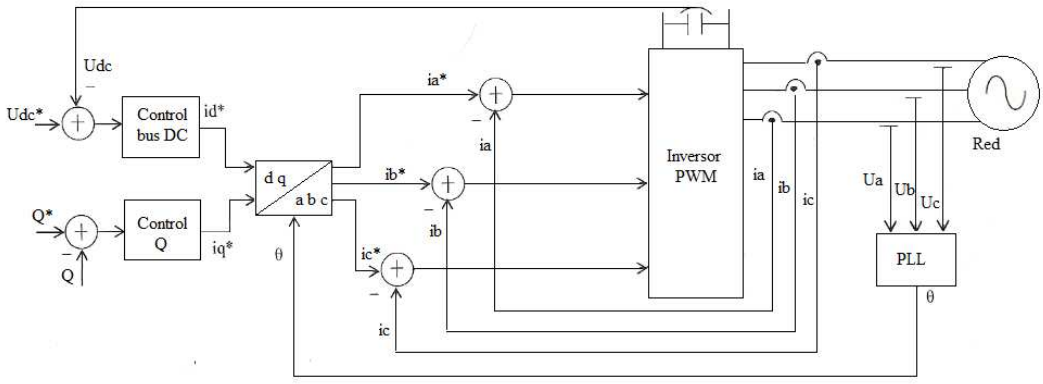


Figura 3 21. Estructura de control de un sistema de control abc.

Mediante la utilización del ángulo de fase de las tensiones de la red suministrados por un sistema PLL, quedan generadas las tres referencias de la corriente. Se compara cada una de ellas con la medida de la corriente correspondiente y el error obtenido va al controlador. En el caso de que se utilice el controlador por histéresis o *"dead-beat"* en el lazo de la corriente, el modulador ya no es necesario.

La salida de estos controladores son los estados de conmutación de los interruptores en el convertidor de potencia. En el caso de que se usen los tres controladores PI o PR, será necesario el modulador para generar los ciclos de trabajo del patrón PWM.

La implementación del controlador PR es sencilla ya que el controlador está en un sistema de referencia estacionario abc. Así mismo, la implementación de los tres controladores es posible.

En la Figura 3.22 se plantea un sistema trifásico usando tres compensadores PI para generar las tensiones de control del inversor PWM trifásico.

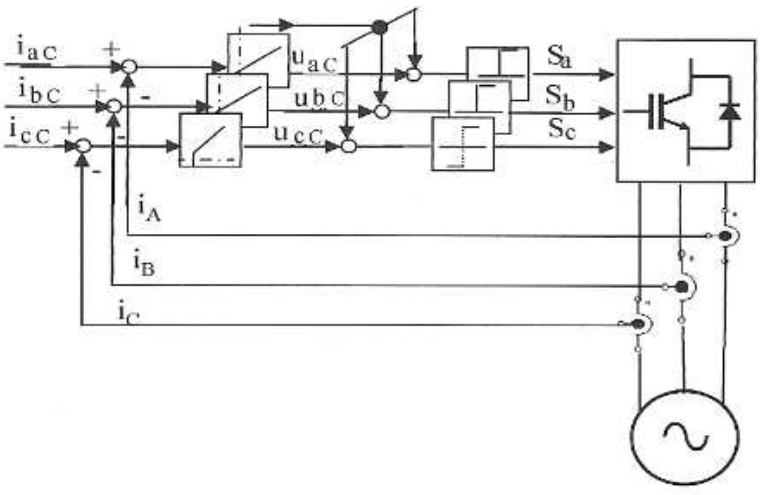


Figura 3 22. Controlador lineal de corriente PI.

Dicho sistema se basa en la comparación de una señal portadora triangular y la moduladora sinusoidal para generar las señales del control de los transistores del inversor. La parte integral del compensador PI minimiza los errores en baja frecuencia, mientras que la parte proporcional de la ganancia y la colocación de los ceros se relacionan con el rizado de la señal.

La amplitud de las señales de control u_{Ac} (u_{Bc}, u_{Cc}), nunca puede ser mayor que la amplitud de la señal triangular. Por otro lado, además de los problemas de cruce de cada triangular, el principal problema de esta técnica es el inherente seguimiento (amplitud y fase) del error.
Con el fin de conseguir la compensación, es recomendable usar un PLL o un *"feed-forward"*.

Hoy en día, el uso de PLL es la técnica más utilizada para la extracción del ángulo de fase de la tensión de la red. Con esto se consigue la sincronización con la tensión de la red.
El PLL se implementa pues en los sistemas de referencia síncronos dq, y su esquema se muestra en la siguiente figura:

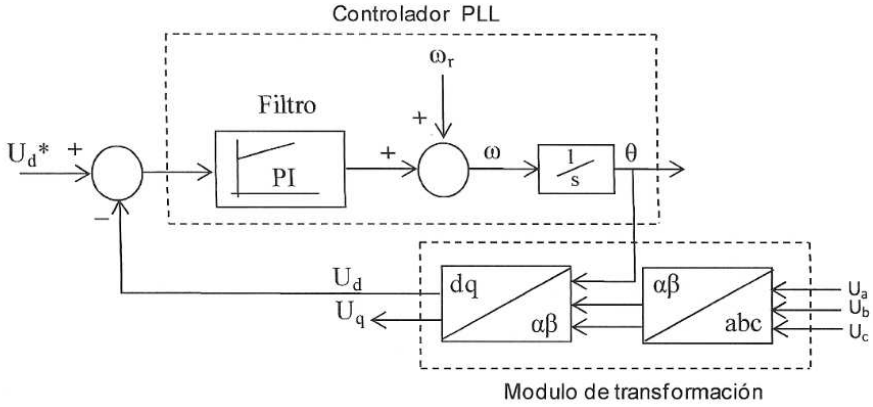


Figura 3 23.Técnica PLL.

Tal y como puede observarse, esta estructura requiere la transformación *abc-dq*. Por lo general, se utiliza un regulador PI para controlar esta variable, y la salida de este regulador es la frecuencia de la red.

Tras la integración de la frecuencia de la red, se obtiene el ángulo de tensión de la red, que se transforma de nuevo del sistema estacionario *αβ* al sistema giratorio de referencia síncrona dq. Este método tiene mejor rechazo de armónicos y de cualquier otro tipo de perturbaciones además permite superar el desequilibrio en la red.

3.1.9 Técnicas de transformación matricial

Los sistemas de potencia han sido analizados tradicionalmente, haciendo uso de matrices de transformación (*Clark, Fortescue, Park,* etc.). Los métodos de representación modal permiten desacoplar las ecuaciones simétricas de los sistemas de potencia. Dichas

técnicas han sido ampliamente usadas para modelar sistemas eléctricos en régimen permanente y para analizar transitorios, la dinámica y los armónicos.

Sistemas simétricos como líneas de transmisión idealmente transpuestas, transformadores y cargas pueden ser desacoplados por alguna transformación modal.

- **Transformación de Fortescue.**

  Este método fue desarrollado en 1918 por D. L. Fortescue en "Método de las coordenadas simétricas" **[15]**, y se aplica a la resolución de redes polifásicas, para soluciones analíticas o analizadores de redes. Sirve para cualquier sistema polifásico desequilibrado: en el cual *n* fasores relacionados entre sí pueden descomponerse en n sistemas de vectores equilibrados (componentes simétricos).

  De acuerdo con el teorema de Fortescue, tres fasores desbalanceados de un sistema trifásico se pueden descomponer en tres sistemas balanceados de fasores. Los conjuntos balanceados de componente son:

  - **Componentes de secuencia positiva** que consisten en tres fasores de igual magnitud desfasados uno de otro por una fase de 120º y que tienen la misma secuencia de fase que las fases originales.
  - **Componentes de secuencia negativa** que consisten en tres fasores iguales en magnitud, desplazados en fase uno de otro en 120º y que tienen una secuencia de fase contraria a las fases originales.
  - **Componentes de secuencia cero (homopolares)** que consisten en tres fasores iguales en magnitud y con un desplazamiento de fase cero uno con respecto al otro.

- **Transformación de *Clarke*.**

  Su objetivo es simplificar un vector formado por tres componentes sinusoidales, las cuales están englobadas dentro de un plano, que esta formado por los ejes *a*, *b* y *c* no ortogonales, transformando este sistema no ortogonal en otro sistema que sí lo es.
  De esta forma, se transforma el vector abc de tres componentes en un vector que tan solo tiene dos componentes, α y β, con valor numérico diferente de cero y que pertenece a un sistema ortogonal **[16]**.

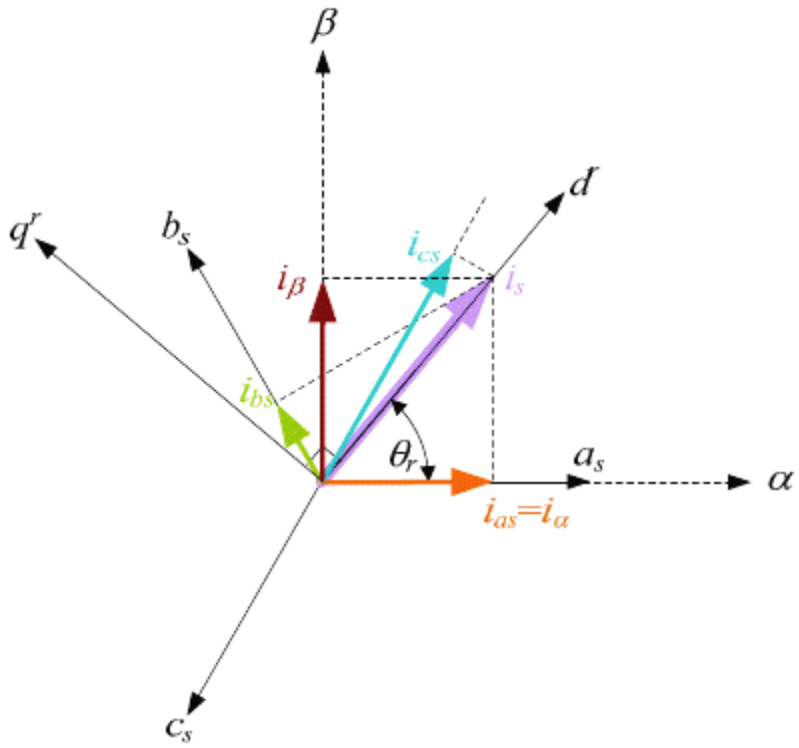


Figura 3 24.Representación del vector I_{abc} y efecto de la transformación de Clarke.

La transformación de *Clarke* se realiza utilizando unas sencillas ecuaciones: que convierten los tres vectores contenidos en un plano, en un vector αβ de tan solo dos componentes, con valor diferente de cero y que pertenece a un sistema ortogonal.

$$i_0 = i_a + i_b + i_c = 0 \qquad (3.3)$$

$$i_\alpha = \frac{2}{3}\left(i_a - \frac{1}{2} i_b - \frac{1}{2} i_c \right) \qquad (3.4)$$

$$i_\beta = \frac{2}{3}\left(\frac{\sqrt{3}}{2} i_b - \frac{\sqrt{3}}{2} i_c \right) \qquad (3.5)$$

$$i_0 = 0 \qquad (3.6)$$

$$i_\alpha = i_a \qquad (3.7)$$

$$i_\beta = \frac{2 \cdot i_b + i_a}{\sqrt{3}} \qquad (3.8)$$

De esta forma, la transformación de *Clarke* permite convertir un sistema trifásico de corrientes, a 120 º en un plano, a un sistema trifásico ortogonal. Además, es una transformación que no modifica el módulo del vector. No obstante, no queda resuelto el problema de la dependencia del ángulo θr.
Esta dependencia puede eliminarse haciendo girar todo el sistema de referencia *α* y *β*, con el ángulo *θr*. Para realizar esto tan solo se ha de conseguir que los ejes de referencia *α* y *β* giren con este valor de ángulo *θr*, y es tan fácil como aplicar la matriz de un giro.

$$\rho(\theta_r) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_r & \sin\theta_r \\ 0 & -\sin\theta_r & \cos\theta_r \end{bmatrix} \qquad (3.9)$$

A modo de resumen, puede concluirse que con la transformación de *Clarke*, se pasa de un sistema trifásico de ondas sinusoidales a un sistema en el plano de ondas sinusoidales y, finalmente, con el giro se consigue que estas ondas se conviertan en rectas y con valores constantes, tal como puede apreciarse en la siguiente figura.

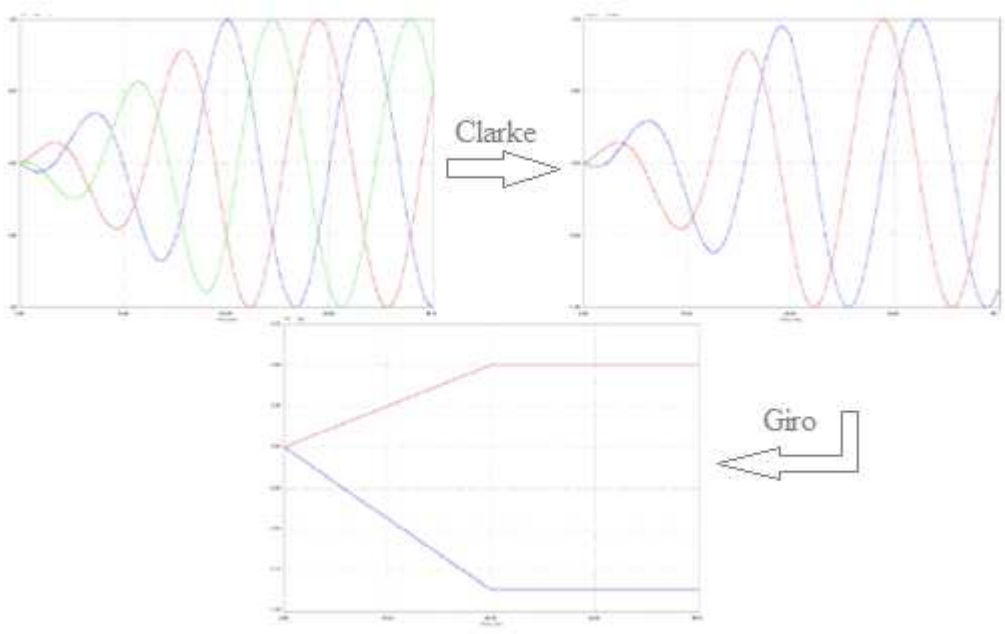


Figura 3 25. Proceso de la transformación matricial de Clarke y la matriz de giro.

- **Transformación de *Park*.**

Así pues, la transformada de *Park* o *dq* es la combinación de la transformada de *Clarke* y el giro, convirtiendo las componentes *abc* del sistema trifásico a otro sistema de referencia *dq0*. El objetivo de la transformación consiste en convertir los valores trifásicos *abc*, variables senoidales en el tiempo, a valores constantes *dq0*, en régimen permanente [**17]**.

El vector con las componentes del nuevo sistema de referencia [x_r] se obtiene multiplicando el vector de coordenadas trifásicas [x] por la matriz de transformación [T], tal como se muestra en la siguiente expresión:

$$\begin{bmatrix} x_d \\ x_q \\ x_0 \end{bmatrix} = [x_r] = [T]\cdot[x] = [T]\cdot\begin{bmatrix} x_a \\ x_b \\ x_c \end{bmatrix} \qquad (3.10)$$

El cambio de nomenclatura tan solo viene a ser un cambio de ejes y un giro, sencillas operaciones algebraicas y geométricas, pero con un claro significado físico, que a continuación se describe.

La expresión de la matriz de transformación [T] se muestra a continuación:

$$[T]=\frac{2}{3}\cdot\begin{bmatrix}\cos(\theta) & \cos\left(\theta-\frac{2\pi}{3}\right) & \cos\left(\theta+\frac{2\pi}{3}\right)\\ -\sin(\theta) & -\sin\left(\theta-\frac{2\pi}{3}\right) & -\sin\left(\theta+\frac{2\pi}{3}\right)\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{bmatrix} \quad (3.11)$$

Donde θ es el ángulo de la referencia rotativa (ejes d-q).

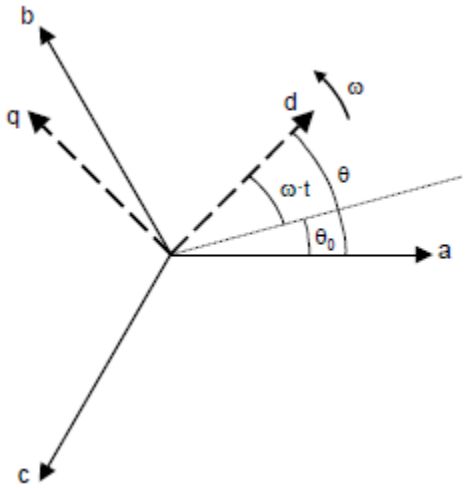


Figura 3 26. Sistemas de referencia trifásicos y d q.

$$\theta=\int_0^t(\omega\cdot t)\cdot dt+\theta_0 \quad (3.12)$$

Donde ω es la velocidad angular de la referencia d-q (igual a la pulsación del sistema trifásico del lado de alterna del convertidor) y θ_0 es el ángulo inicial de la referencia d-q.

Cuando la velocidad angular ω es constante, la transformación se puede expresar según la siguiente ecuación:

$$[T]=\frac{2}{3}\cdot\begin{bmatrix}\cos(\omega\cdot t+\theta_0) & \cos\left(\omega\cdot t+\theta_0-\frac{2\pi}{3}\right) & \cos\left(\omega\cdot t+\theta_0+\frac{2\pi}{3}\right)\\ -\sin(\omega\cdot t+\theta_0) & -\sin\left(\omega\cdot t+\theta_0-\frac{2\pi}{3}\right) & -\sin\left(\omega\cdot t+\theta_0+\frac{2\pi}{3}\right)\\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2}\end{bmatrix} \quad (3.13)$$

El término que multiplica la matriz de transformación puede tener un valor diverso. En las ecuaciones anteriores, este término presenta el valor $\frac{2}{3}$ que será el valor utilizado en PSIM como se verá en el próximo capítulo. No obstante, en

algunas bibliografías se utiliza el valor $\sqrt{\frac{2}{3}}$, ya que con este valor, se consigue que la transformación sea ortonormal, al verificar la propiedad $[T]^{-1} = \sqrt{\frac{2}{3}} \cdot T^{T}$, tal como se muestra a continuación:

$$[T] = \sqrt{\frac{2}{3}} \cdot T \Rightarrow [T]^T = \sqrt{\frac{2}{3}} \cdot T^T \qquad (3.14)$$

$$[T] \cdot [T]^T = \sqrt{\frac{2}{3}} \cdot T \cdot \sqrt{\frac{2}{3}} \cdot T^T = \frac{2}{3} \cdot T \cdot T^T = [I]_{3x3} \Rightarrow [T]^T = [T]^{-1} \qquad (3.15)$$

Las transformaciones ortonormales se caracterizan por mantener invariante el producto escalar:

$$[x_{1r}] = [T] \cdot [x_1] \quad ; \quad [x_{2r}] = [T] \cdot [x_2]; \qquad (3.16)$$

$$[x_{1r}]^T \cdot [x_{2r}] = ([T] \cdot [x_1])^T \cdot ([T] \cdot [x_2]) = [x_1]^T \cdot [T]^T \cdot [T] \cdot [x_2] = [x_1]^T \cdot [x_2] \qquad (3.17)$$

Como consecuencia de la anterior propiedad, el valor de la potencia instantánea se mantiene invariante, independientemente del dominio donde se calcule *abc* o *dq0*.

$$[v_{fr}] = [T] \cdot [v_f] \quad ; \quad [i_{fr}] = [T] \cdot [i_f] \qquad (3.18)$$

$$p = v_a \cdot i_a + v_b \cdot i_a + v_c \cdot i_c = [v_a \quad v_b \quad v_c] \cdot \begin{bmatrix} i_a \\ i_b \\ i_c \end{bmatrix} = [v_f]^T \cdot [i_f] \qquad (3.19)$$

$$p = [v_f]^T \cdot [i_f] = ([v_f]^T \cdot [T]^T) \cdot ([T] \cdot [i_f]) = [v_{fr}]^T \cdot [i_{fr}] \qquad (3.20)$$

$$p = [v_{fr}]^T \cdot [i_{fr}] = [v_d \quad v_q \quad v_0] \cdot \begin{bmatrix} i_d \\ i_q \\ i_0 \end{bmatrix} = v_d \cdot i_d + v_q \cdot i_q + v_0 \cdot i_0 \qquad (3.21)$$

Según las características del sistema trifásico y a partir de las expresiones de T y de las componentes de la denominada secuencia homopolar o cero, pueden realizarse las siguientes deducciones:

- Si el sistema trifásico de tensiones está equilibrado, la suma de tensiones abc es nula en todo momento ($v_a+v_b+v_c=0$) y la tensión homopolar (v_0) es nula.
- Si el neutro del sistema trifásico está aislado, la suma de corrientes abc es nula en todo momento ($i_a+i_b+i_c=0$) y la corriente homopolar (i_0) es nula.
- Si el sistema trifásico de tensiones está equilibrado y la impedancia de carga es la misma en todas las fases (carga equilibrada), las sumas de tensiones y corrientes abc son nulas en todo momento y las componentes a secuencia cero son nulas (v_0, $i_0=0$).

Suele ser común que el sistema trifásico de tensiones sea simétrico y equilibrado, y que la carga trifásica esté equilibrada. En estas condiciones, las componentes homopolares son nulas y la aplicación de la transformación de *Park* o d-q reduce el número de variables del sistema, al pasar de tres variables trifásicas abc a dos variables dq de valor constante en régimen permanente.

3.2 Lenguajes de descripción hardware

3.2.1 Introducción

Dentro de los lenguajes de descripción de hardware, están ampliamente extendidos *"Verilog"* y *"VHDL" ("Very High Speed Integrated Circuit Hardware Description Language").*

Este último ha tenido mayor difusión debido principalmente a su normalización con un estándar del IEEE, lo cual motivó a muchas empresas y centros de investigación a utilizarlo. Cada vez son más los diseñadores que utilizan este lenguaje para describir los circuitos integrados digitales.

Como otra alternativa también puede tenerse en cuenta VHDL-AMS (*"Analog and Mixed Signal extensions"*), que se trata de la extensión analógica del VHDL y permite simular sistemas analógicos y mixtos, junto al VHDL tradicional. Para crear los modelos de convertidores, este lenguaje incluye medios para desarrollarlos de una forma más fácil que escribiéndolos mediante VHDL. No obstante, las herramientas de simulación de VHDL-AMS están aún en desarrollo, por lo que se descarta esta opción, que por otra parte, parece la elección natural para resolver el problema planteado.

A continuación se hace una breve introducción al lenguaje de descripción hardware que se ha utilizado para llevar a cabo este proyecto.

3.2.2 Lenguaje VHDL

Definición

VHDL (VHSIC (*"Very High Speed Integrated Circuit"*) *"Hardware Description Language"*) es un lenguaje de descripción hardware creado para modelar y simular sistemas digitales en distintos niveles de abstracción **[18]**.

Historia

A mediados de los años setenta se produce una fuerte evolución en los procesos de fabricación de los circuitos integrados, y junto a las tecnologías bipolares, surge la MOS (*"Metal Oxide Semiconductor"*), principalmente la NMOS (*"Negative Channel Metal*

Oxide Semiconductor"), promoviendo el desarrollo de circuitos digitales hasta la primera mitad de los años ochenta.

En aquella época, el esfuerzo de diseño se concentraba en los niveles eléctricos para establecer características e interconexiones entre los componentes básicos a nivel de transistor.

El proceso de diseño era altamente manual y únicamente se empleaban herramientas como PSPICE para simular esquemas eléctricos con modelos previamente personalizados a las distintas tecnologías.

Con el paso de los años, los procesos tecnológicos se hacían más y más complejos. Los problemas de integración iban en aumento y los diseños eran cada vez más difíciles de depurar y de dar mantenimiento. Inicialmente los circuitos MSI ("*Medium Scale Integration"*) y LSI ("*Low Scale Integration"*) se diseñaron mediante la realización de prototipos basados en módulos más sencillos. Cada uno de estos módulos estaba formado por puertas ya probadas y este método fue quedándose obsoleto poco a poco. En ese momento (finales de los años setenta) se constata el enorme desfase que existe entre tecnología y diseño.

La considerable complejidad de los chips que se pueden fabricar, implica unos riesgos y costes de diseño desmesurados e imposibles de asumir por las empresas. Es entonces, cuando diversos grupos de investigadores empiezan a crear y desarrollar los llamados "lenguajes de descripción de hardware" cada uno con sus peculiaridades. Empresas tales como IBM con IDL ("*Interface Description Language"*), TI -HDL de *Texas Instruments*, ZEUS de *General Electric*, etc., así como los primeros prototipos empleados en las universidades, empezaron a desarrollarse buscando una solución a los problemas que presentaba el diseño de los sistemas complejos.

Sin embargo, estos lenguajes nunca alcanzaron el nivel de difusión y consolidación necesarias por varios motivos. Los lenguajes industriales, por ser propiedad de la empresa permanecieron encerrados en ellas y no estuvieron disponibles para su estandarización y mayor difusión y por otro lado, los lenguajes universitarios, perecieron por no disponer de soporte ni mantenimiento adecuado.
Alrededor de 1981, el Departamento de Defensa de los Estados Unidos desarrolla un proyecto llamado VHSIC ("*Very High Speed Integrated Circuit"*) cuyo objetivo era rentabilizar las inversiones en hardware haciendo más sencillo su mantenimiento.
Se pretendía con ello resolver el problema de modificar el hardware diseñado en un proyecto para utilizarlo en otro, lo que no era posible hasta entonces porque no existía una herramienta adecuada que armonizase y normalizase dicha tarea, era el momento de los HDL's.

En 1983, *IBM*, *Intermetrics* y *Texas Instruments* empezaron a trabajar en el desarrollo de un lenguaje de diseño que permitiera la estandarización, facilitando con ello, el mantenimiento de los diseños y la depuración de los algoritmos, para ello el IEEE propuso su estándar en 1984.

Tras varias versiones llevadas a cabo con la colaboración de la industria y de las universidades, que constituyeron a posteriori etapas intermedias en el desarrollo del lenguaje, el IEEE publicó en diciembre de 1987 el estándar IEEE std 1076-1987 que constituyó el punto firme de partida de lo que después de cinco años sería conocido como VHDL.

Esta doble influencia, tanto de la empresa como de la universidad, hizo que el estándar asumido fuera un compromiso intermedio entre los lenguajes que ya habían desarrollado previamente los fabricantes, de manera que este quedó como ensamblado y por consiguiente un tanto limitado en su facilidad de utilización haciendo dificultosa su total comprensión.

Pero esta deficiencia se ve altamente recompensada por la disponibilidad pública, y la seguridad que le otorga el verse revisada y sometida a mantenimiento por el IEEE.

La independencia en la metodología de diseño, su capacidad descriptiva en múltiples dominios y niveles de abstracción, su versatilidad para la descripción de sistemas complejos, su posibilidad de reutilización y la independencia de que goza con respecto de los fabricantes, han hecho que VHDL se convierta con el paso del tiempo en el lenguaje de descripción de hardware por excelencia.

Descripción

El lenguaje VHDL nació para dar respuesta a numerosos problemas planteados en el desarrollo y documentación de hardware digital. La documentación que se requiere para describir un sistema electrónico suele ser de muy costoso reemplazamiento cuando la tecnología o las especificaciones cambian, por lo que un lenguaje de descripción adecuado resuelve el problema al ser dicha documentación ejecutable.

VHDL se ha convertido en un estándar, que ya es en sí una ventaja, pero además reúne otras características que proporcionan ventajas muy significativas. Se trata de un lenguaje independiente de la tecnología, que no se encuentra ligado a un determinado simulador y que no requiere una metodología precisa de diseño. Además, VHDL permite implementar nuevas tecnologías en diseños existentes.

Se trata de un lenguaje de semántica orientada a la simulación y por ello su principal dominio de aplicación es el modelado de dispositivos hardware para comprobar su correcto funcionamiento. Pero también tiene otras áreas de aplicación como son la síntesis automática, verificación formal, modelado de rendimiento, diagnosis de fallos y documentación.

El lenguaje de descripción de hardware VHDL cuenta con diferentes modos de llevar a cabo la descripción. La descripción estructural resalta la correspondencia existente entre realidad y lenguaje, pero oculta la verdadera potencia del modelo temporal soportado por este lenguaje. La descripción comportamental y la ejecución concurrente de procesos ponen de manifiesto la verdadera potencia de VHDL.

Cada circuito a implementar puede ser visto como una caja negra que se relaciona con el exterior mediante un conjunto de señales, unas de entrada y otras de salida. Esta caja

negra se conoce en VHDL como entidad. A cada entidad le corresponde al menos uno de los siguientes modos de descripción que permite VHDL:

- **Comportamental:** En cuyo modelado, lo importante es la función que relaciona la salida con la entrada. De esta forma, se permite escribir funciones complejas sin recurrir a su implantación física, proporcionando una potencia de diseño muy atractiva a muy bajo coste.

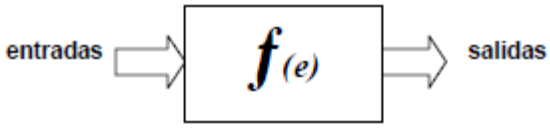


Figura 3 27. Esquema de modelado comportamental.

- **Estructural:** En este segundo modo de descripción, se especifican los bloques que componen un circuito y sus interconexiones. Cada uno de sus bloques integrantes debe poseer su descripción previa de forma que se construya una jerarquía de descripciones donde las inferiores dan lugar a superiores más complejas.

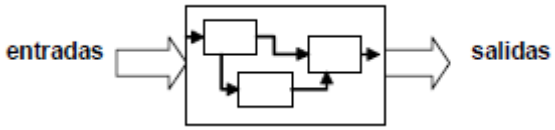


Figura 3 28. Esquema de modelado estructural.

- **RTL (*Resistor Transfer Logic*) o modelado de flujo de datos:** En este último modo de descripción se declara la sucesión temporal con la que evolucionan las diferentes señales del modelo descrito.

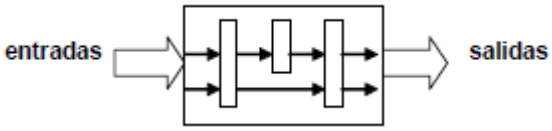


Figura 3 29. Esquema de modelado RTL.

Cada uno de estos modos de descripción lleva asociada una determinada sintaxis que lo caracteriza. A continuación se muestran brevemente los elementos básicos de sintaxis VHDL:

- La declaración de entidad se realiza mediante la palabra reservada "ENTITY" seguida de una etiqueta identificadora. Dentro de la entidad se declaran las señales de entrada y salida, genéricos y constantes.
- La declaración de la arquitectura utiliza la palabra "ARCHITECTURE" seguida de su identificador y de identificador de la entidad a la que se asocia y con la que deberá ser consistente en tipos y nombres de señales. Inmediatamente después se encuentra la parte declarativa de la arquitectura que sirve para declarar variables y señales locales a la arquitectura.
- Después de "BEGIN" se encuentran las sentencias concurrentes del modelo que se ejecutarán de manera simultánea en tiempo de simulación, de manera que cada una de ellas parte del mismo instante de simulación.

- Un proceso marca los límites del dominio secuencial y sólo se evalúa si cambia alguna de las señales de su lista de sensibilización, esto es aquella lista proporcionada entre paréntesis después de la palabra “PROCESS”. Cada sentencia secuencial partirá del origen de tiempos que marque el final de la sentencia anterior.

Referente al modelo temporal en VHDL, es importante destacar la importancia de distinguir entre los tiempos físicos y los de simulación.

Como último punto en esta breve descripción del lenguaje VHDL, y debido a la importancia que ha cobrado a lo largo del desarrollo de este proyecto, es necesario mencionar un tipo de descripción a caballo entre la estructural y la comportamental y que se utiliza para comprobar el correcto funcionamiento de los modelos mediante un programa independiente de la plataforma de compilación-simulación.

Dicho tipo de descripción se conoce como *“test-bench”* y no es más que una entidad sin puertos. La arquitectura cuenta con una instancia del modelo bajo test y un proceso que le inyecta señales. Cuando se simla un *“test-bench”*, se obtiene un cronograma de respuesta a estas señales de entrada.

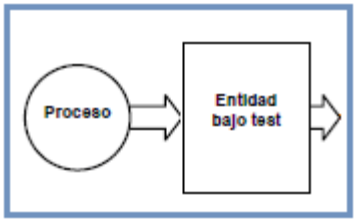


Figura 3 30 Esquema de “test-bench”.

3.3 Herramientas de desarrollo

En este capítulo, se va a hacer una descripción de las herramientas de desarrollo empleadas a lo largo de este Proyecto Final de Carrera para poder llevar a cabo las diferentes simulaciones.

3.3.1 Modelsim

ModelSim es una herramienta de simulación y depuración para diseños en VHDL ampliamente extendida. Es un proceso de diseño asistido por computador, en el que se utiliza una metodología descendente que se compone de las siguientes fases:

- **Especificación:**
  Donde se establece la descripción de la funcionalidad del circuito.
- **Diseño arquitectural:**

Consiste en obtener una descripción sintetizable del circuito que cumpla las especificaciones, así como los bancos de prueba que permitan comprobar su correcto funcionamiento mediante simulación.
El resultado de esta fase es una síntesis de alto nivel y una simulación de carácter funcional.

- **Diseño detallado:**
  Utilizando una herramienta de síntesis automática, se obtiene la lista de puertas (“*netlist”*), proporcionada por el fabricante de los circuitos en los que se va a implementar el diseño. Modificaciones en el diseño pueden mejorar los resultados de esta fase, ya sea optimizando el área o en tiempo. Cabe la posibilidad de realizar una simulación de post-síntesis.
- **Diseño físico:**
  Consiste en el emplazamiento y rutado dentro del dispositivo a programar. Así mismo, se validan los retardos del circuito. Opcionalmente se podrán realizar simulaciones post-rutado.
- **Programación y pruebas:**
  Se utiliza la herramienta de programación del fabricante para implementar el diseño arquitectural

La herramienta ModelSim **[19]** como simulador VHDL permite generar el comportamiento de los elementos internos y las salidas, tomando como entrada el circuito descrito mediante este lenguaje y los estímulos a las entradas,

La validación funcional a este nivel consiste en comprobar si el comportamiento descrito mediante un lenguaje de descripción hardware (VHDL) coincide con las especificaciones del diseño.

Esta comprobación se lleva a cabo utilizando un modelo funcional de los componentes que contiene la descripción y un conjunto de estímulos de entrada que se aplican a ese modelo. Mediante un computador, se “ejecuta” ese modelo aplicándole los estímulos y se obtienen las salidas del diseño, este proceso se denomina simulación.

Modelsim permite el diseño, compilación y simulación de diseños digitales realizados con VHDL, ofreciendo funciones útiles para la depuración. La interfaz de usuario resulta potente y eficaz para manejar diseños grandes; permitiendo, entre otras ventajas, ficheros de comandos que se ejecutan como una macro.

A partir de bibliotecas propias o genéricas, paquetes definidos por el usuario y los archivos de diseño, se consigue una compilación del diseño que permite la validación funcional de este en el ordenador. Dicha validación pasa por la generación de un banco de pruebas en el que se instancia el diseño y se le aplican los estímulos necesarios para validar su funcionamiento.

3.3.2 PSIM

Para desarrollar el control digital en un DSP, se necesita simular el código de los mismos con el modelo de circuito de potencia a controlar. Este es un proceso de diseño muy utilizado y sistematizado. Dos de las posibilidades más usadas son:

- **Matlab/Simulink:** Altamente integrada con el desarrollo y probablemente la más empleada.
- **PSIM:** Herramienta de simulación fundamentalmente analógica que incorpora una biblioteca de bloques de control digital tales como funciones de transferencia en z, biestables, puertas lógicas, muestreadores, comparadores, etc. También es usada por un gran número de diseñadores.

La herramienta de simulación analógica elegida es PSIM **[20]**, ya que este paquete de software está específicamente diseñado para el diseño de convertidores de potencia y circuitos de control.

PSIM contiene un conjunto de programas que permiten capturar los esquemáticos de los circuitos junto con su correspondiente sistema de control (continuo o discreto) y realizar simulaciones en el dominio del tiempo de los sistemas diseñados.

Las ventajas de este simulador frente a simuladores basados en el clásico Pspice son:

- La alta velocidad de simulación.
- El enfoque a la simulación de sistemas electrónicos de potencia y su correspondiente sistema de control (ya sea digital, analógico o mixto), presentando una interfaz de usuario adecuada para estos fines.
- La convergencia está prácticamente siempre asegurada, ya que se suelen simular dispositivos ideales o casi ideales.

El paquete de simulación PSIM consiste en tres programas integrados en el flujo de diseño, tal y como se muestra en la Figura 3.31.

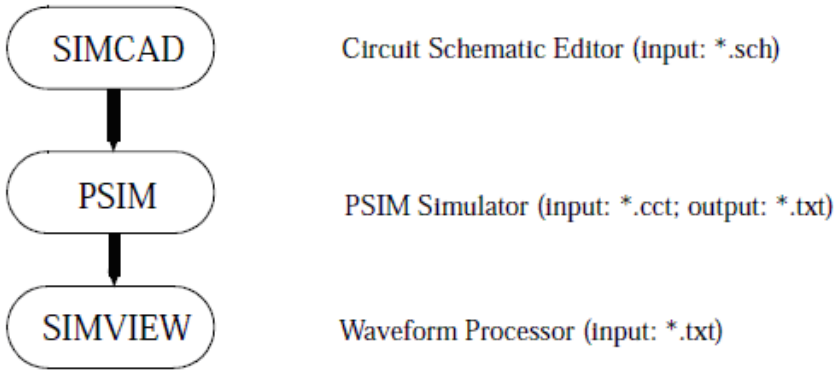


Figura 3 31 Flujo de diseño en PSIM.

Cabe destacar que el *"Circuit Schematic Editor"* permite introducir los esquemas de los circuitos y definir las excitaciones aplicadas, mientras que el programa *"SIMVIEW"* permite visualizar gráficamente los resultados de las simulaciones. Estas simulaciones se obtienen del programa de simulación eléctrico que es transparente al usuario.

El componente principal a utilizar en la aplicación propuesta en este Proyecto es el "*dll_block*", que se trata de un bloque que llama a una *dll* (*"Dynamic Linking Library"*) creada por el usuario. Estas bibliotecas de enlace dinámico son archivos con código ejecutable que se cargan bajo demanda.

Este bloque, que deberá ser incluido en el esquemático, se presenta con varias configuraciones de pines: 1, 3, 6, 12, 20 o 25 entradas y salidas. El nombre de la *dll* se fija dentro del bloque, y el fichero debe encontrarse en el mismo directorio que el esquemático.

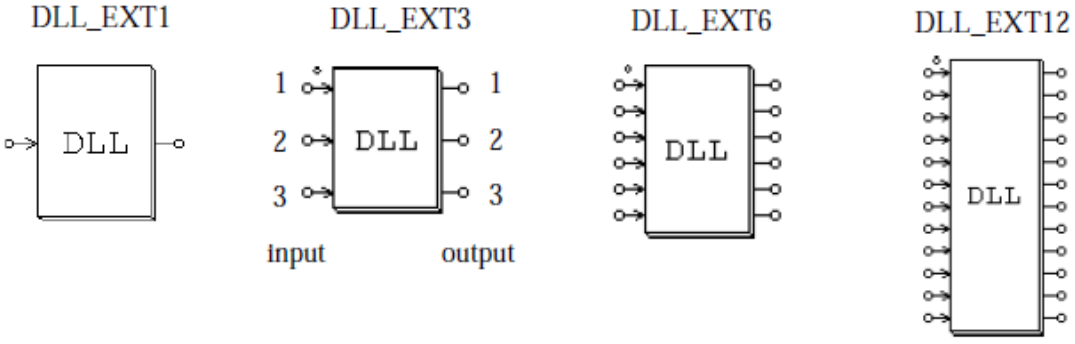


Figura 3 32 Figura dll_block.

Un bloque *dll* recibe los valores de PSIM como entrada, realiza los cálculos, y envía el resultado de vuelta a PSIM. PSIM llama a la rutina *dll* en cada tiempo de simulación. Sin embargo, cuando las entradas del bloque *dll* se conectan a cualquiera de los elementos discretos disponibles, el bloque *dll* será llamado únicamente en los instantes discretos de muestreo.

Ya sea en tiempo de simulación o de muestreo cuando se llama a la *dll*, esta adquiere los valores de las entradas, realiza los cálculos definidos en su código y fija las salidas. Una vez fijadas las salidas, la simulación continúa hasta el siguiente instante de llamada.

Cabe destacar por otro lado, que las entradas que no sean utilizadas deben ponerse a tierra.

3.4 Metodologías disponibles

La integración de simulaciones que se va a presentar, lleva implícito el desarrollo de una metodología de diseño de convertidores de potencia con control digital. Las fases del diseño de un sistema electrónico de potencia con control digital, son las siguientes:

- Objetivos y especificaciones del sistema.
- Diseño del circuito de potencia.
- Diseño del control digital.
- Integración de los diseños.
- Construcción de prototipo y pruebas experimentales.

Para el diseño de convertidores de potencia, se siguen las etapas que a continuación se describen:

- **Definición de objetivos y especificaciones.** A partir de la definición básica del diseño, se concretan los detalles del diseño.
- **Diseño de la etapa de potencia**. Incluye la selección de los valores de los componentes a partir de las especificaciones. Esta etapa también incluye el modelado de los componentes y la simulación de la etapa de potencia, haciendo uso de estos modelos.
- **Diseño del lazo de control de la etapa de potencia**. En esta etapa se fija la respuesta dinámica del convertidor, la estabilidad y la variación de la tensión de salida, en lazo cerrado. La validación del diseño se realiza simulando la etapa de potencia y el bucle de control, usando para ello la herramienta PSIM.
- **Prototipado del convertidor en lazo cerrado y pruebas experimentales.**

Para el diseño de circuitos integrados digitales existen dos metodologías de diseño de circuitos integrados, la ascendente (*"bottom-up"*) y la descendente (*"top-down"*). Esta última es, por su eficacia, la más extendida en la actualidad. Ambas se diferencian en el nivel de abstracción del que se parte para describir el circuito.
A continuación se recuerdan los distintos niveles de abstracción y los estilos de descripción que puede tener la representación de un circuito:

1. Nivel eléctrico.
2. Nivel de transistores.
3. Nivel de puertas lógicas.
4. Nivel de transferencia de registros.
5. Nivel algorítmico.
6. Nivel de sistema.

Por otro lado, en cada uno de estos niveles de abstracción se puede describir el circuito mediante tres estilos de descripción, según se describa el comportamiento, la estructura o la implementación física. El estilo de comportamiento consiste en describir el circuito por lo que hace, la "función" que expresa las salidas en función de las entradas y sus estados internos. El estilo de estructura describe el circuito con los elementos que lo componen, los bloques internos y su comunicación. Y finalmente, el estilo físico describe el circuito tal y como es, por ejemplo, en el nivel de transistores es el dado de silicio resultante de la fabricación, en el nivel de sistema es la placa con sus componentes y conexiones.

En la siguiente figura, se muestra la relación entre niveles de abstracción y estilos de descripción dentro del diseño, propuesta por "*Gajski"*.

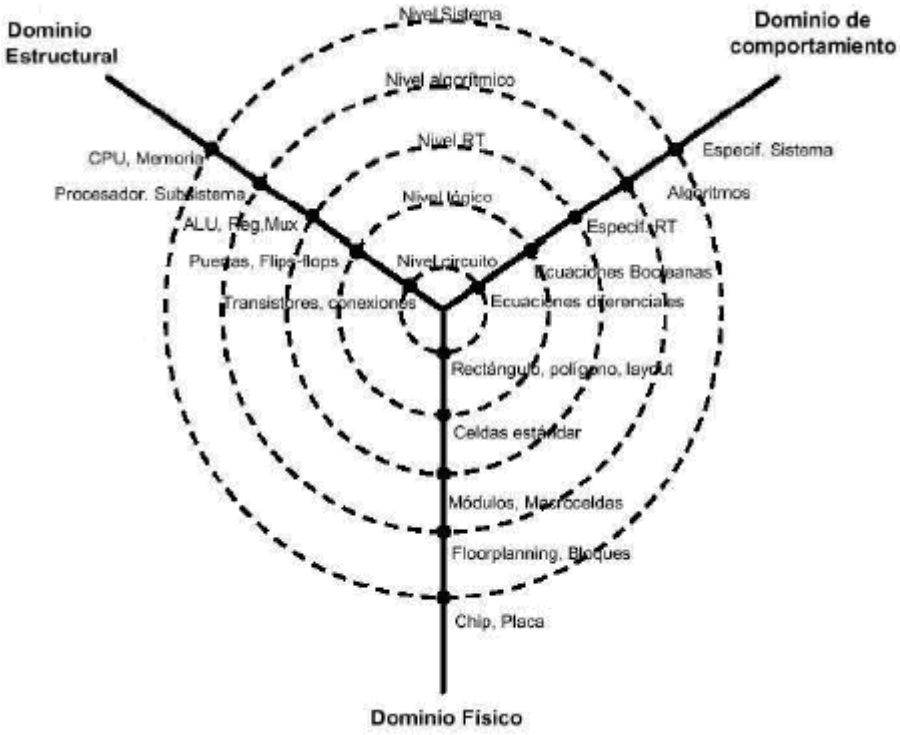


Figura 3 33 Diagrama de Y de Gajski- Khun. Niveles de abstracción y dominios descriptivos

Cada una de las ramas equivale a un estilo de descripción y cada círculo se corresponde con un nivel de abstracción.
El diseño se realiza avanzando desde los extremos hacia el centro, pasando de los niveles de abstracción superiores (más externos) a los inferiores (cercanos al centro), y cambiando de estilo de descripción en un mismo nivel de abstracción cambiando de rama en el círculo correspondiente.

En la **metodología ascendente**, la etapa de diseño comienza con la división en bloques funcionales del circuito. Dichos bloques funcionales se dividen en tres sub-bloques más pequeños, y así constantemente hasta llegar a obtener unidades funcionalmente sencillas de implementar. Mediante esta topología se diseña cada parte de forma independiente y, una vez realizadas, se procede a conectarlas para formar el circuito completo.

En la **metodología descendente,** se parte del mismo punto que en la ascendente, no obstante la forma de obtener el circuito final es diferente. En este caso, se desciende en nivel de abstracción desde las especificaciones hasta el circuito físico. De esta forma, se propone una forma de diseñar los circuitos en un nivel de abstracción más alto, empleando los lenguajes de descripción de hardware para describir la funcionalidad. Así se tiene una descripción funcionalmente completa del circuito global desde el primer momento, tal como se muestra en la Figura 3.34.

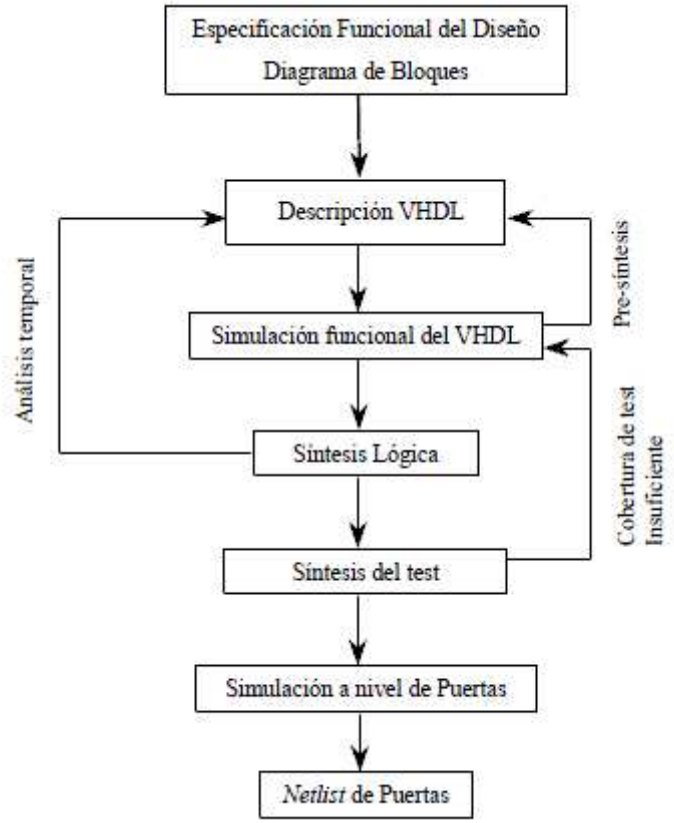


Figura 3 34 Esquema genérico del flujo de diseño descendente.

Como puede verse, a partir de la idea o concepto que se desea implementar en forma de sistema o circuito electrónico se ha de proceder a una primera fase de definición de especificaciones. Hasta ahora esta ha sido la etapa menos formalizada y a la que no se presta la atención que requiere, a pesar de que los resultados de la misma guían o dirigen muchas de las decisiones posteriores durante el proceso de diseño. Con la llegada de los HDLs se estableció la tecnología de base para dar el soporte necesario a la actividad de concepción al nivel funcional.

Una vez fijadas las especificaciones, comienza la descripción HDL de las mismas mediante un proceso de refinamiento gradual de la descripción del circuito hasta alcanzar un modelo arquitectural-RT que sea sintetizable mediante procesos automáticos guiados por el diseñador.

A continuación viene la etapa de síntesis lógica que tiene por objeto la obtención de un esquema o lista de componentes y sus interconexiones basado en una determinada biblioteca de celdas y módulos. El diseño resultante debe realizar la funcionalidad especificada, respetar las prestaciones requeridas (área, velocidad, consumo) y garantizar la testabilidad final del circuito.

Capítulo 4

Descripción general de la aplicación

A lo largo de este capítulo se describe el diseño de un inversor cuyo control será implementado de forma digital, y descrito mediante el lenguaje VHDL, prestando principal atención en el sistema de control del inversor, ya que sobre él se centran las principales aportaciones en el circuito presentado.

El control del inversor diseñado y desarrollado mediante módulos digitales en este Proyecto Final de Carrera, sirve como prueba de concepto de las posibilidades del control digital, así como de la metodología de diseño mixto utilizada. El paso a una implementación hardware en un circuito integrado (ASIC) o en un dispositivo programable (FPGA) no forma parte de los objetivos de este proyecto y su consecución abre nuevos trabajos futuros.

4.1 Modelo del inversor trifásico con control *dq*

Uno de los objetivos que debe cumplir un inversor de potencia es el de controlar la potencia que se inyecta a la red, de acuerdo a las normativas.

Con el fin de realizar mejoras significativas en el diseño y la aplicación práctica de los inversores, deben tenerse en cuenta aspectos como la reducción de la distorsión armónica,

cancelación de la componente continua de la corriente inyectada, control del factor de potencia o el uso de control digital.

Otro punto a tener en cuenta en los inversores conectados a la red, es el tipo de potencia que se inyecta (activa o reactiva) en base a la cual modificar el carácter del factor de potencia. Aquellos sistemas más eficientes son los que permiten modificar tanto la potencia activa (aquella que es disipada en forma de calor o trabajo), como la reactiva (aquella utilizada para la formación de los campos eléctrico y magnético) inyectada a la red.

En la siguiente figura se muestra el esquema completo del inversor trifásico con control *dq*, que posteriormente se quiere diseñar con módulos digitales (e implementar en VHDL), tal y como se explicará en los dos capítulos siguientes.

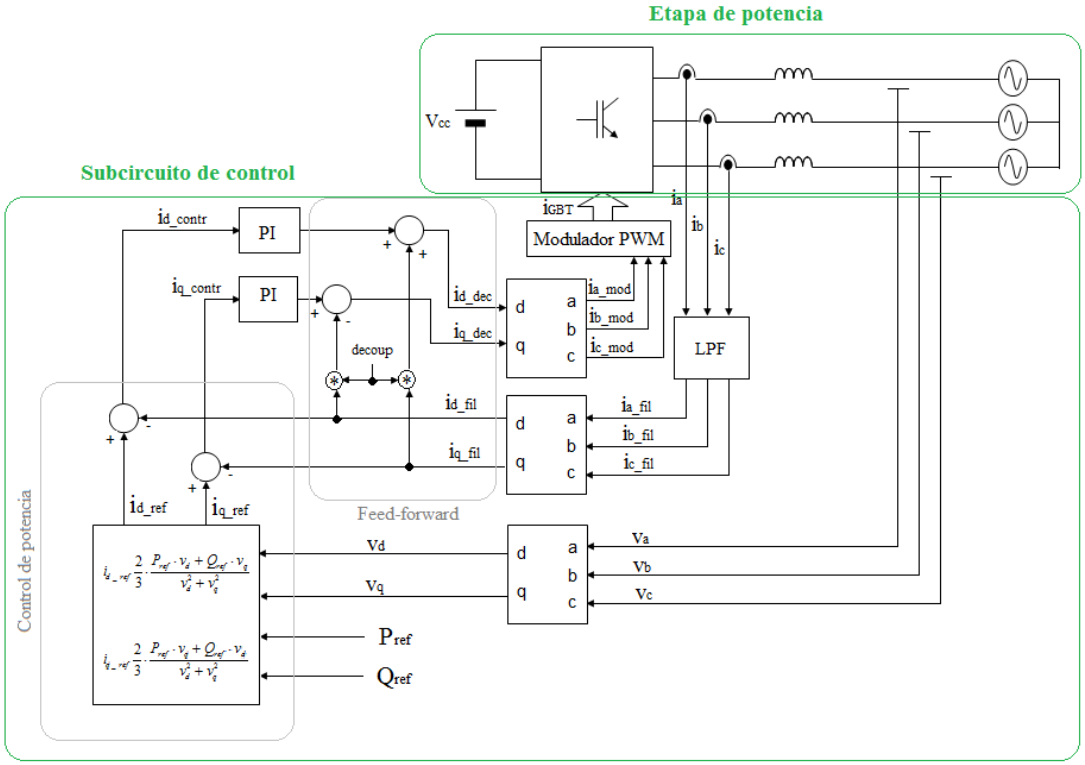


Figura 4.1. Diagrama de bloques del sistema inversor conectado a red.

Como puede observarse en la Figura 4.1, el sistema inversor planteado consta de los siguientes bloques:

- **Etapa de potencia** que realiza la conversión de continua a alterna.
- **Subcircuito de control** compuesto por:
  - **Filtro paso bajo (LP)** aplicado sobre las corrientes de línea para eliminar el rizado existente en ellas.
  - **Transformada directa de *Park*** para la generación de componentes *dq* tanto de la tensión de red como de la corriente de línea. Mediante la transformada de Park, las señales variables en el tiempo *abc*, pasan a encontrarse en otro sistema de referencia con valores constantes *dq0*, en régimen permanente.

- **Control de potencia** activa y reactiva mediante la generación de corrientes de referencia a partir de las tensiones de red en régimen permanente.
- **Regulador PI** que permitirá alcanzar un nuevo régimen permanente admisible ante una perturbación, y de la manera más rápida posible.
- **Red *"Feed-forward"*** para lograr el desacoplo en las coordenadas *dq* debido al efecto de las bobinas. En el diagrama mostrado, este efecto se indica mediante la señal *decoup* que multiplica a i_{d_fil} e i_{q_fil}.
- **Transformada inversa de *Park*** que una vez realizada la compensación de potencia y eliminado el acoplo existente entre las señales *d* y *q* permite volver al dominio *abc* antes de entrar al bloque de modulación PWM.
- **Modulación PWM** que permite controlar la energía proveniente de las señales periódicas obtenidas i_{a_mod}, i_{b_mod} *e* i_{c_mod} que serán transmitidas finalmente a la red.

A continuación se presentan en detalle cada uno de los bloques que componen el modelo inversor propuesto.

4.2 Etapa de potencia

El inversor trifásico requiere pocos requisitos de filtrado para la reducción de armónicos. Además, como se verá más adelante, el control de la amplitud se produce a la frecuencia fundamental.
A continuación se muestra el esquemático en PSIM de este bloque.

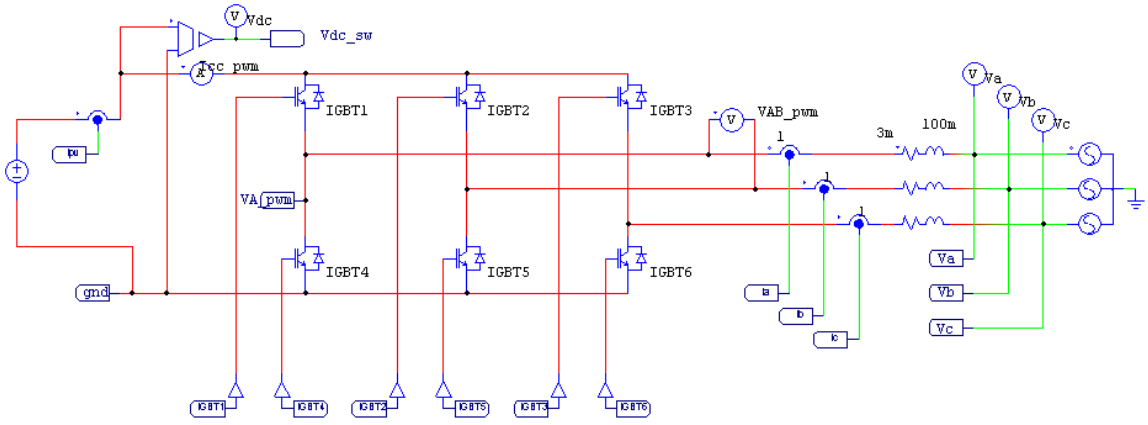


Figura 4.2. Etapa de potencia del inversor trifásico.

La conmutación, tal y como se verá en el apartado 4.9 de este capítulo, se lleva a cabo comparando cada una de las tres ondas senoidales obtenidas tras la modulación PWM y desplazadas entre sí 120º, con una misma portadora triangular. La amplitud de salida vendrá determinada por la relación entre la portadora y la moduladora.

Tal como puede apreciarse en la Figura 4.2, los interruptores controlados se han llevado a cabo mediante transistores IGBT, que combinan las ventajas de los transistores bipolares

(baja tensión de saturación) con las de transistores MOSFET (alta impedancia de entrada). Estos dispositivos son adecuados para este montaje, ya que la tensión de continua es elevada (500 V) y la frecuencia de conmutación se encuentra por debajo de 40kHz (1500 Hz como se verá más adelante).
Por otro lado, es necesario integrar un transistor junto con un diodo para lograr bidireccionalidad en corriente y unidireccionalidad en tensión.

Se trata de un inversor autoconmutado que se puede conectar a la red ya que es posible sincronizar su tensión alterna de salida con la tensión de la red eléctrica, de manera que se puede inyectar cualquier nivel de corriente a la red. La regulación de tensión se realiza mediante modulación del ancho del pulso PWM, aumentando el ancho de los pulsos que disparan los interruptores de potencia en el caso de que la tensión a la salida disminuya y viceversa. Este inversor es capaz de conmutar a altas frecuencias y puede generar la señal de corriente en fase con la señal de tensión de la red, corrigiendo el factor de potencia.

La tensión de salida de cada una de las patas del inversor, por ejemplo la tensión V_{A_pwm} de la pata A, únicamente depende de la tensión continua de entrada V_D y del estado de los interruptores de esa pata, siendo independiente de la intensidad de salida si se considera que siempre se encontrará activado uno de los interruptores, esto es, se desprecia el *"blanking time"* o tiempo muerto y se asume la idealidad de los interruptores.

A continuación se muestra el ciclo de trabajo PWM centrado en una de las señales trifásicas en coordenadas *abc* que se inyectan a este primer bloque del inversor.
La siguiente figura muestra el resultado de la señal *a* modulada.

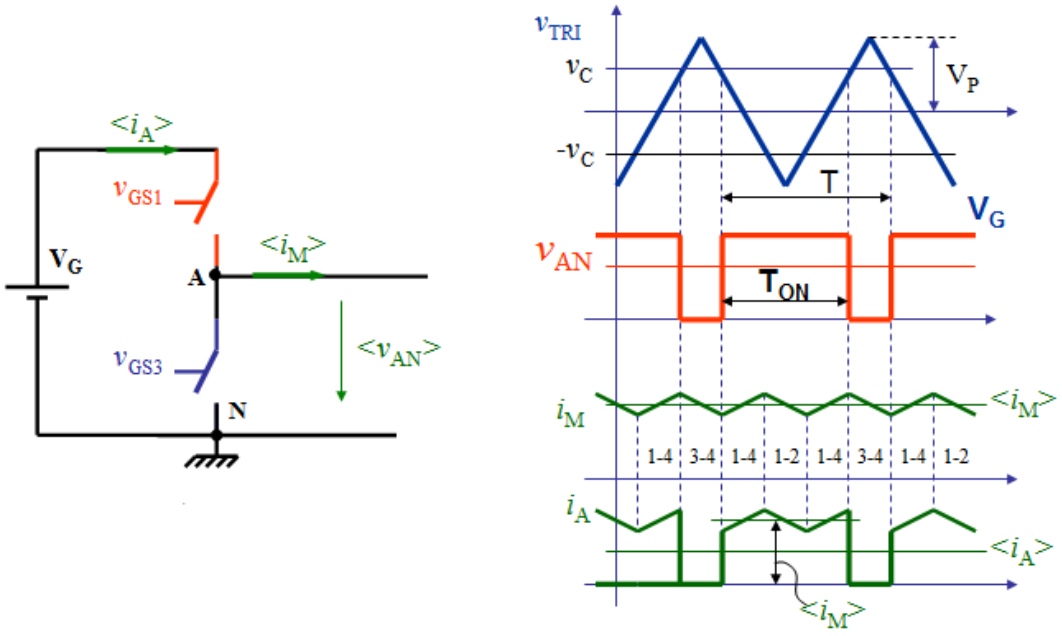


Figura 4.3.Señal modulada mediante PWM.

Los parámetros fundamentales del PWM son el periodo *T* y el ciclo de trabajo *D*, que indica el tiempo que la función toma valor uno respecto al tiempo total (periodo).

$$D = \frac{T_{ON}}{T} \qquad (4.1)$$

De acuerdo con la Figura 4.3, el ciclo de trabajo para la señal *a* en particular, es el siguiente:

$$D_{GS1} = D_A = \frac{1}{2} \cdot \left[1 + \frac{v_C}{V_p} \right] \tag{4.2}$$

De esta forma, la corriente que finalmente se inyecta a la red (i_A) puede representarse de la siguiente forma:

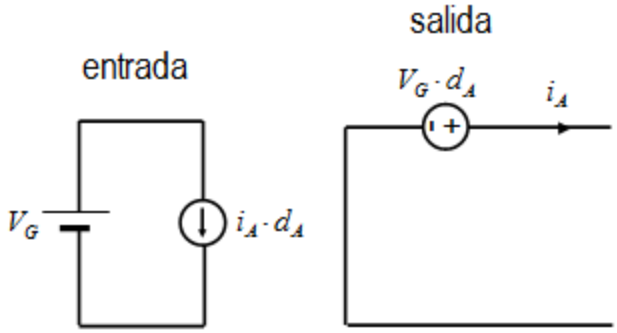


Figura 4.4. Modelo promediado monofásico.

Siendo:

$$\langle V_{AN} \rangle = V_G \cdot D_A \tag{4.3}$$

$$\langle i_A \rangle = \langle i_M \rangle \cdot D_A \tag{4.4}$$

De aquí en adelante, se usará la siguiente notación:

i_A: Valor promediado.
$\hat{i}_A$: Pequeña señal.
I_A: Punto de trabajo.

En la siguiente figura se muestra el modelo promediado en coordenadas *abc* de la conexión a red de las tres ramas del inversor a la frecuencia de conmutación:

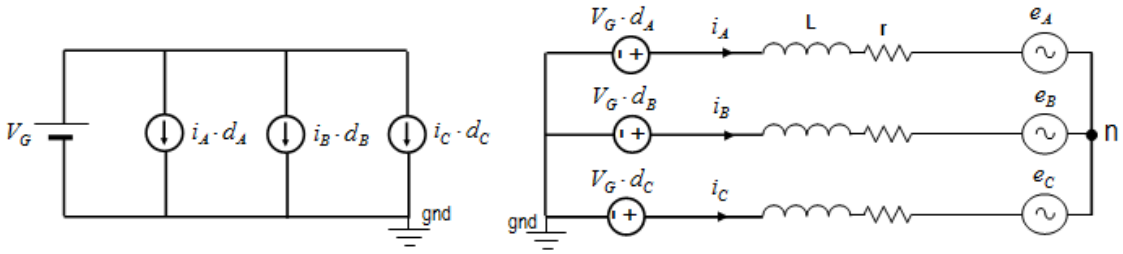


Figura 4.5.Modelo promediado trifásico de la conexión a red.

Así pues, de acuerdo con la Figura 4.5, se obtiene la siguiente expresión para la corriente de línea:

$$L \cdot \frac{\partial i_A}{\partial t} = V_G \cdot d_A - r \cdot i_A - e_A \tag{4.5}$$

$$L \cdot \frac{\partial i_B}{\partial t} = V_G \cdot d_B - r \cdot i_B - e_B \tag{4.6}$$

$$L \cdot \frac{\partial i_C}{\partial t} = V_G \cdot d_C - r \cdot i_C - e_C \qquad (4.7)$$

Finalmente:

$$\frac{\partial}{\partial t}\begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} = \frac{V_G}{L} \cdot \begin{bmatrix} d_A \\ d_B \\ d_C \end{bmatrix} - \frac{r}{L} \cdot \begin{bmatrix} i_A \\ i_B \\ i_C \end{bmatrix} - \frac{1}{L} \cdot \begin{bmatrix} e_A \\ e_B \\ e_C \end{bmatrix} \qquad (4.8)$$

4.3 Subcircuito de control

A lo largo de este apartado se analizarán cada uno de los bloques que componen el subcircuito de control del sistema inversor propuesto.

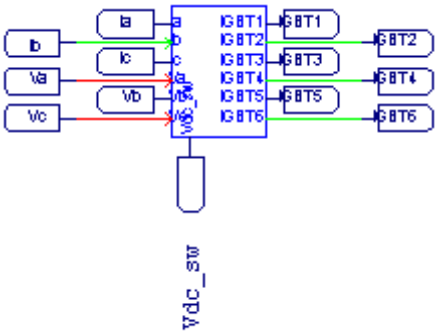


Figura 4.6.Subcircuito de control.

A lo largo de este capítulo se han ido mostrando cada uno de los bloques que componen el subcircuito de control: filtro paso bajo, transformadas directa e inversa de *Park*, control de potencia, reguladores PI, red "*feed-forward*" y modulación PWM.
En la siguiente figura se muestra el circuito completo correspondiente al subcircuito de control analógico explicado:

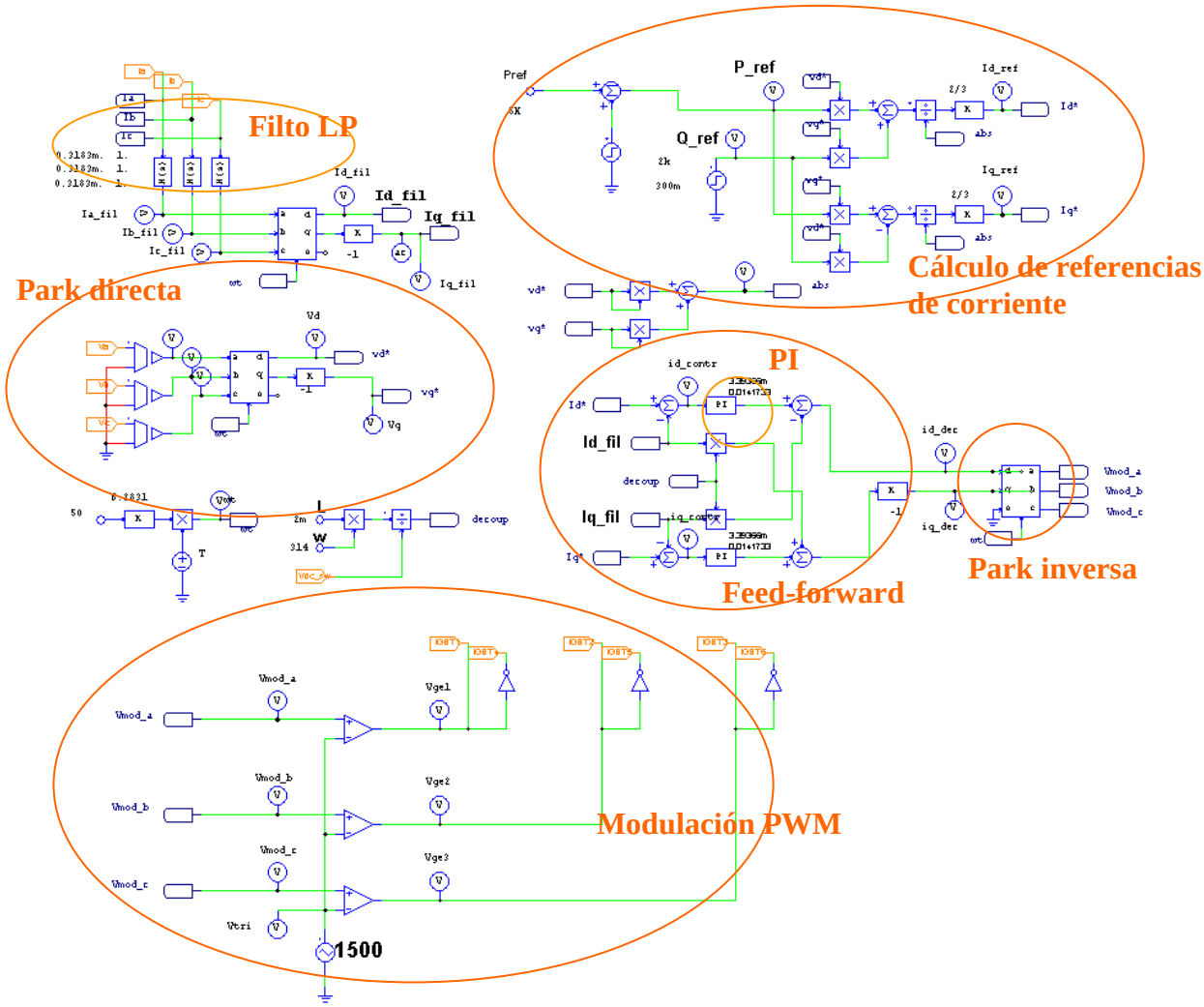


Figura 4.7. Bloques del subcircuito de control.

4.3.1 Filtro paso bajo (LP)

Debido al rizado presente en la corriente de línea, se aplica un filtro paso bajo sobre cada una de las tres corrientes de línea desfasadas 120º, tal como se muestra en la Figura 4.8.

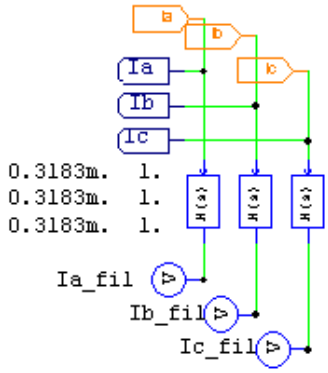


Figura 4.8.Filtrado de la corriente de red.

Se trata de un filtro de orden uno y ganancia también la unidad, cuya función de transferencia en el dominio de Laplace viene dado por:

$$G(s) = 1 \cdot \frac{1}{0.0003183 \cdot s + 1} \qquad (4.9)$$

A continuación se muestran las señales de entrada y salida obtenidas para este bloque:

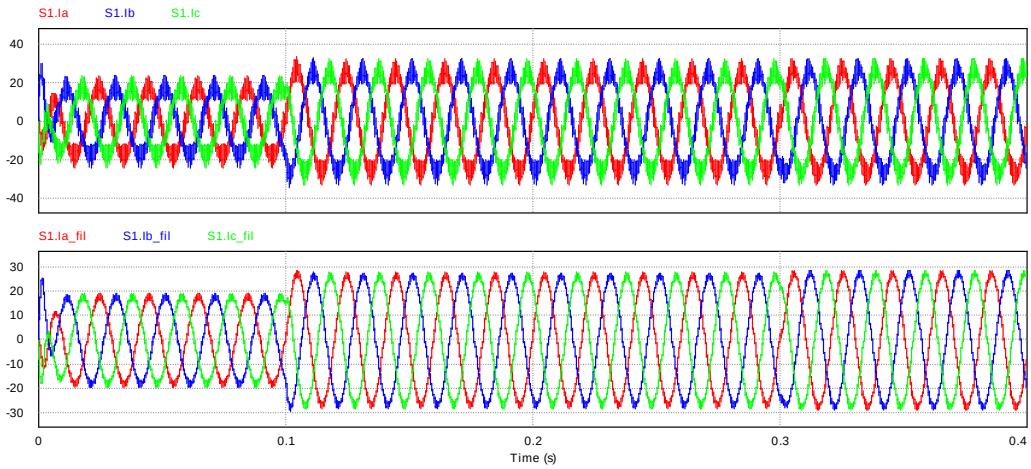


Figura 4.9.Corrientes de red tras el filtrado paso bajo.

4.3.2 Transformada directa de *Park*

Como ya se ha comentado previamente, en el circuito planteado a lo largo de este Proyecto se aplica la transformada de *Park* tanto sobre las corrientes de red una vez filtradas paso bajo como sobre las tensiones de red, tal y como se muestra en la siguiente figura:

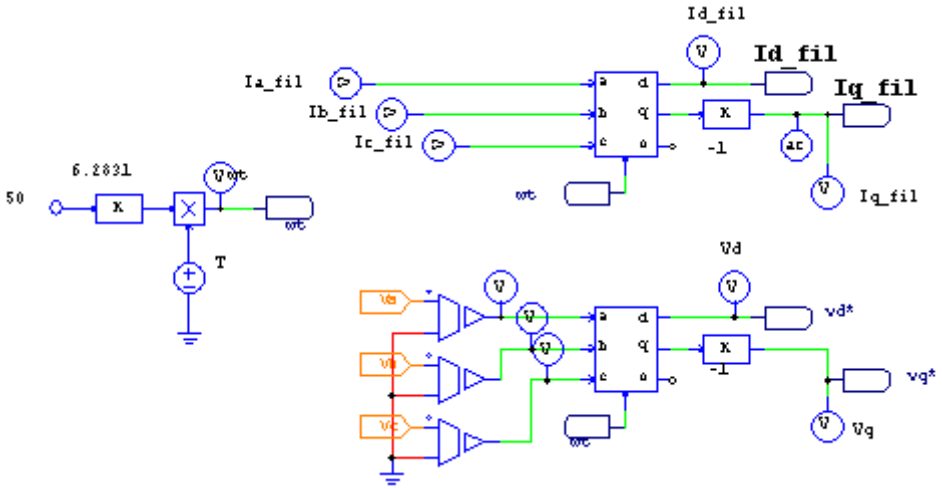


Figura 4.10.Generación de componentes dq.

Cabe notar en la Figura 4.10, que el circuito que genera la señal *wt* permite asumir que la fase de red comienza en el instante *t = 0*.

En inversores que emplean esquemas de modulación basados en PWM, como es el caso que aplica, el ruido en la tensión de salida se produce idealmente alrededor de la frecuencia de conmutación y de sus múltiplos. Sin embargo, los tiempos muertos,

necesarios para evitar cortocircuitos en la parte de continua del inversor producen armónicos no deseados de baja frecuencia en la tensión de salida del inversor.

Para mejorar el contenido armónico de la tensión de salida se emplea la transformada de *Park* para pasar a un eje de referencia síncrono, ya que en este nuevo sistema de referencia, la acción de controles es muy sencilla de entender al ser todas las variables magnitudes continuas en régimen permanente.

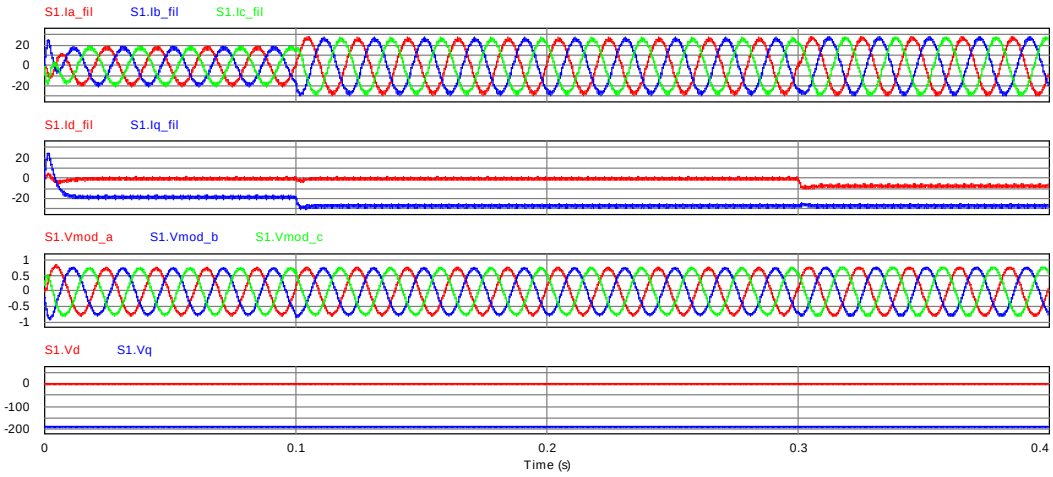


Figura 4.11.Componentes dq obtenidos en tensión y corriente.

La transformada directa de *Park* aplicada en el sistema inversor propuesto se expresa como:

$$T_{abc\,/\,dqo} = \frac{2}{3}\cdot\begin{bmatrix} \cos(w\cdot t) & \cos\left(w\cdot t-\frac{2\pi}{3}\right) & \cos\left(w\cdot t+\frac{2\pi}{3}\right) \\ -sen(w\cdot t) & -sen\left(w\cdot t-\frac{2\pi}{3}\right) & -sen\left(w\cdot t+\frac{2\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad (4.10)$$

Con: *a,b,c*: Arrollamiento trifásico general giratorio a la velocidad *w* (fases reales).
d,q,o: Arrollamiento ortogonal giratorio a w_a (*q* adelanta a *d*).
θ_1: Ángulo eléctrico entre *a* y *d*.

Por otra parte, la expresión de la transformada inversa de *Park* puede expresarse como se indica:

$$T_{dqo\,/\,abc} = \begin{bmatrix} \cos(w\cdot t) & -sen(w\cdot t) & 1 \\ \cos\left(w\cdot t-\frac{2\pi}{3}\right) & -sen\left(w\cdot t-\frac{2\pi}{3}\right) & 1 \\ \cos\left(w\cdot t+\frac{2\pi}{3}\right) & -sen\left(w\cdot t+\frac{2\pi}{3}\right) & 1 \end{bmatrix} \qquad (4.11)$$

Por tanto:

$$[I_{abc}] = T_{dqo\,/\,abc}\cdot[I_{dqo}] \qquad (4.12)$$

$$[D_{abc}] = T_{dqo/abc} \cdot [D_{dqo}] \quad (4.13)$$

$$[E_{abc}] = T_{dqo/abc} \cdot [E_{dqo}] \quad (4.14)$$

La técnica *"feed-forward"* logra que el ciclo de trabajo tenga las dependencias contrarias que posteriormente incluirá la planta (etapa de potencia).
No obstante, será necesario diseñar controles PI para anular el error en régimen permanente entre la referencia y la salida, como se verá más adelante.

4.3.3 Cálculo de referencias de corriente

Del mayor o menor retraso o adelanto que provoque un equipo eléctrico cualquiera en la corriente que fluye por un circuito, en relación con el voltaje o tensión, así será el factor de potencia o cos (φ) que tenga dicho equipo.
En un circuito eléctrico de corriente alterna se pueden llegar a encontrar tres tipos de potencias eléctricas diferentes:

- **Potencia activa (P) (resistiva).** Trabajo útil que genera una carga resistiva conectada a un circuito de corriente alterna. Se expresa como:
  $P=V \cdot I \cdot cos\ (\varphi)$ *(4.15)*
  En los dispositivos que poseen solamente carga resistiva, el factor de potencia es siempre igual a "1", mientras que en los que poseen carga inductiva ese valor será siempre menor de "1".
- **Potencia reactiva (Q) (inductiva).** No proporciona ningún tipo de trabajo útil, pero los dispositivos que poseen enrollados de alambre de cobre, requieren ese tipo de potencia para poder producir el campo magnético con el cual funcionan. Viene dada por:
  $Q = \sqrt{S^2 - P^2}$ *(4.16)*
- **Potencia aparente (S) (total).** Es el resultado de la suma geométrica de las potencias activa y reactiva. Esta potencia es la que realmente suministra una planta eléctrica cuando se encuentra funcionando al vacío, es decir, sin ningún tipo de carga conectada. Puede escribirse como:
  $S=V \cdot I$ *(4.17)*

Tanto la potencia instantánea activa como la reactiva, están dotadas de sentido físico. La potencia real describe el flujo de energía por unidad de tiempo intercambiado entre dos subsistemas, mientras que la potencia imaginaria es un intercambio energético por unidad de tiempo entre las distintas fases del sistema, pero sin intercambio de energía neto.

Actuando adecuadamente sobre *P* y *Q* se puede acondicionar la potencia del sistema desde el punto de vista de la mitigación de armónicos y/o la compensación de potencia activa y reactiva.

Así pues, mediante el siguiente circuito se obtienen las referencias de corriente para el regulador PI partiendo de los valores de potencia activa y reactiva ($P_{\_ref}$ y Q_{ref}) y de las referencias de tensión (v_d y v_q), que son las tensiones de red en un eje de referencia síncrono.

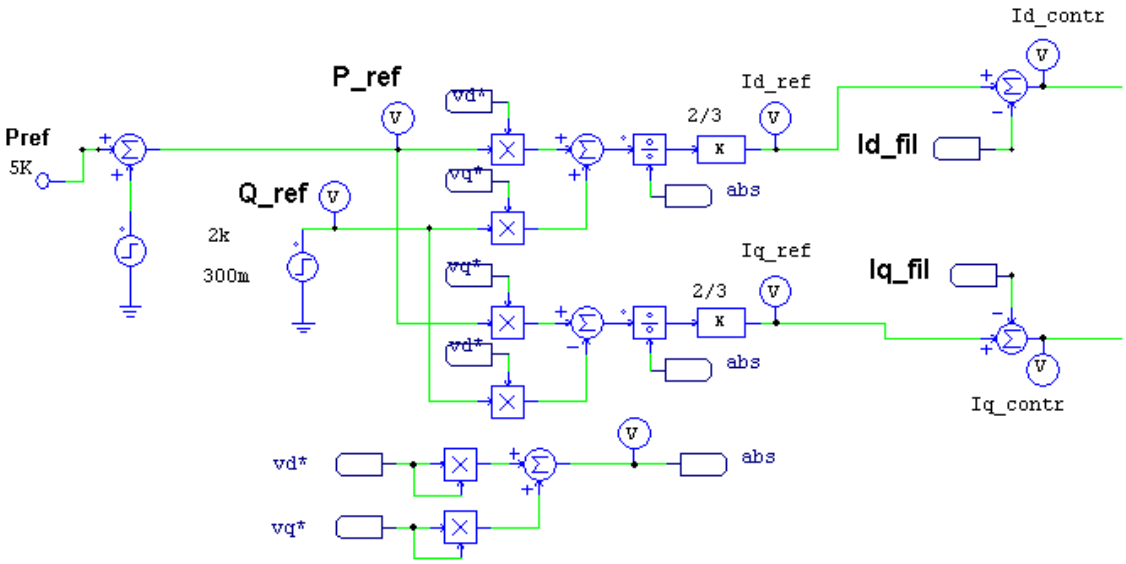


Figura 4.12.Control de potencia.

Tal como podía avanzarse en la Figura 4.1, las señales de referencia generadas en este bloque se obtienen de la siguiente manera:

$$i_{d\_ref} = \frac{2}{3} \cdot \frac{P_{ref} \cdot v_d + Q_{ref} \cdot v_q}{v_d^2 + v_q^2} \qquad (4.18)$$

$$i_{q\_ref} = \frac{2}{3} \cdot \frac{P_{ref} \cdot v_q + Q_{ref} \cdot v_d}{v_d^2 + v_q^2} \qquad (4.19)$$

A continuación se muestran los valores de las potencias activa y reactiva que entran en juego en el circuito estudiado, así como la referencia de corriente en coordenadas *dq* obtenidos en este bloque, y las referencias I_{d_ref} e I_{q_ref} que utilizará el regulador PI para compensar las potencias.

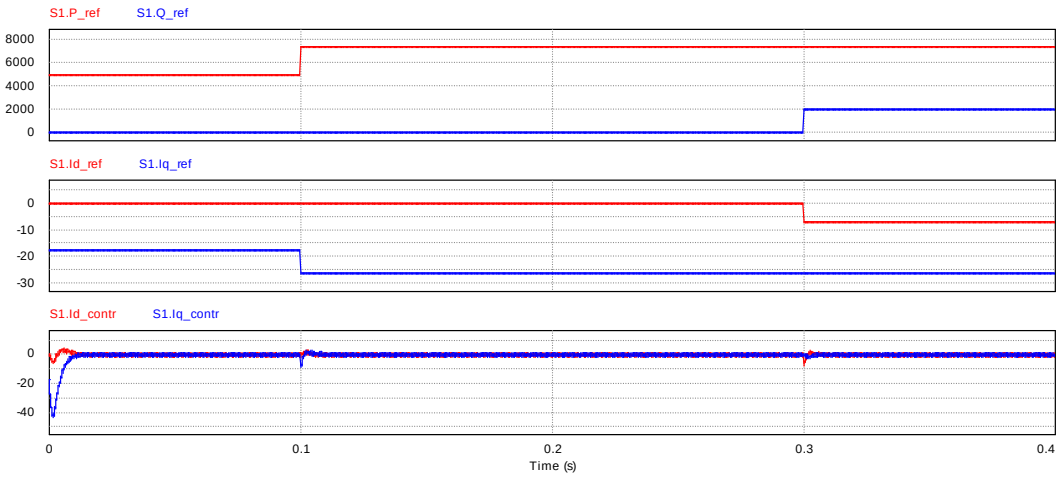


Figura 4.13. Relación de entradas/salidas del bloque de control de potencia.

4.3.4 Regulador PI

La calidad de un sistema de control viene dada por el comportamiento del sistema tanto en régimen permanente como en transitorio. Por lo tanto, los principales puntos a tener en cuenta en el diseño de un sistema de control son:

- Error del sistema (desviación entre la variable controlada y la referencia) ha de ser cercano a cero en el caso estacionario y en presencia de todas las perturbaciones.
- El sistema ha de ser estable, es decir, deberá alcanzar un nuevo régimen permanente con la mayor rapidez posible.

A continuación se calculan los requisitos del regulador en módulo y fase, teniendo en cuenta que el margen de fase es la frecuencia de cruce f_C:

Criterio de fase:

$$\angle Ti_d(w_c) = -180 + MF = arctg\left(Ki_{PI} \cdot w_c\right) - arctg\left(\frac{L}{r} \cdot w_c\right) - 90 \qquad (4.20)$$

$$arctg\left(Ki_{PI} \cdot w_c\right) = -90 + MF - arctg\left(\frac{L}{r} \cdot w_c\right) = \psi \qquad (4.21)$$

$$Ki_{PI} = \frac{tg\,\psi}{w_c}; \psi = -90 + MF - arctg\left(\frac{L}{r} \cdot w_c\right) \qquad (4.22)$$

Donde Ki_{PI} es la constante integral del regulador, L y r son la inductancia y resistencia asociadas a la bobina y MF el margen de fase que se quiere dejar.

Criterio de módulo:

$$\left|Ti_d(w_{c)}\right| = 1 \qquad (4.23)$$

$$K_{SC} \cdot V_G \cdot \frac{1}{\sqrt{r^2 + (Lw_c)^2}} \cdot Kp_{PI} \cdot \frac{\sqrt{1 + (tg\,\psi)^2}}{tg\,\psi} = 1 \qquad (4.24)$$

$$Kp_{PI} = \frac{tg\,\psi}{\sqrt{1 + (tg\,\psi)^2}} \cdot \frac{1}{K_{SC}} \cdot \frac{\sqrt{r^2 + (L \cdot w_c)^2}}{V_G} \qquad (4.25)$$

Siendo Kp_{PI} la constante proporcional del regulador.

Así, la función de transferencia para el regulador PI será:

$$PI(s) = Kp_{PI} \frac{1 + Ki_{PI} \cdot s}{Ki_{PI} \cdot s} \qquad (4.26)$$

La función de transferencia del regulador PI empleado en PSIM responde a la siguiente ecuación:

$$PI(s) = 0.0141733 \cdot \frac{1 + 0.00339366 \cdot s}{0.00339366 \cdot s} \qquad (4.27)$$

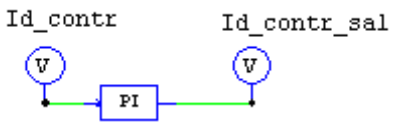


Figura 4.14. Regulador PI.

En las siguientes figuras se muestra el diagrama de bode en amplitud y en fase del lazo de corriente en lazo abierto, representado por su función de transferencia.

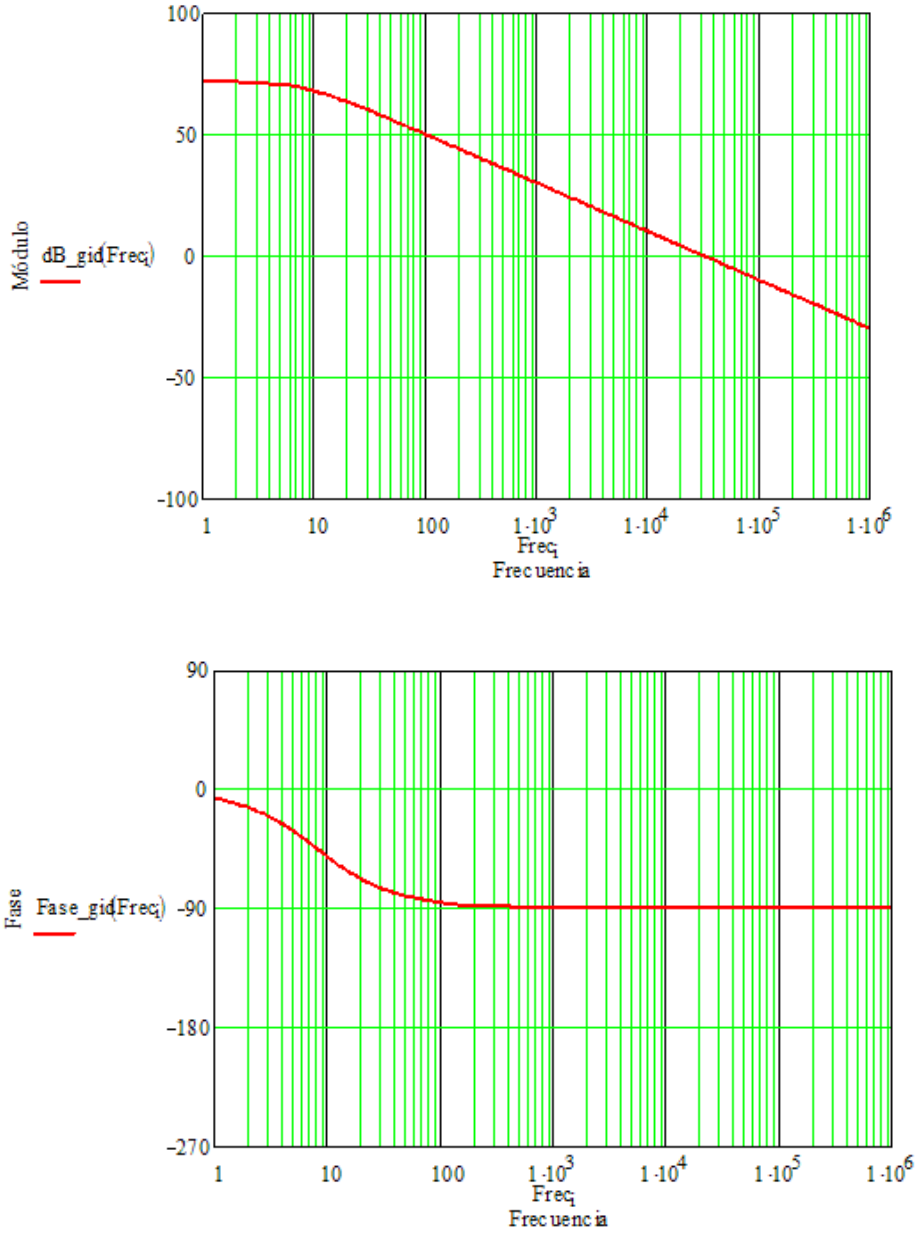


Figura 4.15. Diagrama de Bode en lazo abierto.

Para el circuito a implementar, los valores de cada uno de los parámetros a tener en cuenta son:

$L=3\cdot 10^{-3}$ H; Inductancia.
$r=100\cdot 10^{-3}$ Ω; Resistencia serie de la inductancia.
$f_{sw}=1500$; Frecuencia de conmutación
$K_{SC}=1$; Ganancia del sensor de corriente
$V_G=500$; Tensión en la batería en vacío.

Para garantizar un margen de fase suficiente, la curva de ganancia debe cruzar el eje 0 dB con una pendiente de -20dB/dec.

El margen de fase se encontrará entre el rango comprendido de 30º a 70º, siendo el margen de fase especificado para este lazo de 60º.Los requisitos de lazo abierto serán:
La frecuencia de corte se asume mucho menor que la frecuencia de conmutación que $f_C<f_{conmutación}/6$, tomando así $f_C=200$.

En la Figura 4.13 se aprecia la mejoría obtenida a la salida del regulador, observando un pronto alcance del régimen permanente ante posibles perturbaciones.

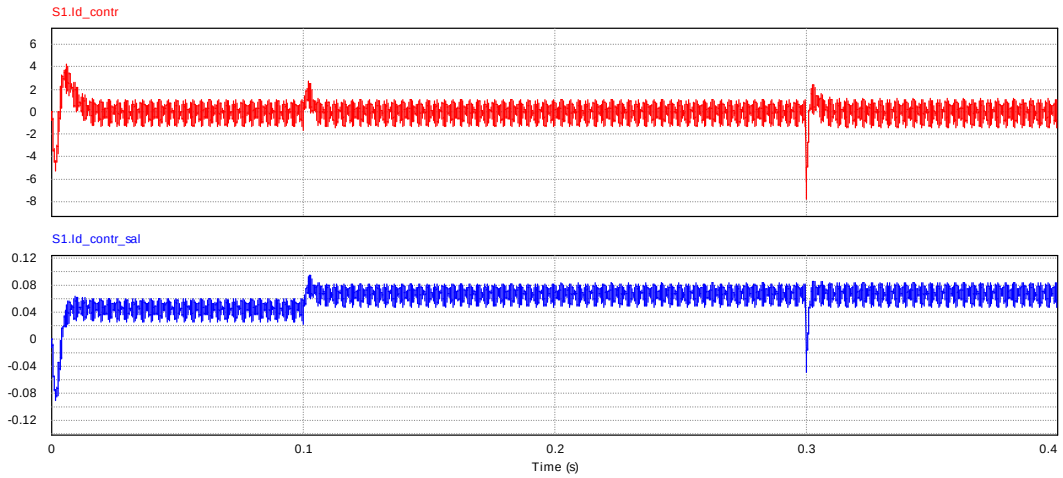


Figura 4.16. Entrada/salida del regulador PI.

4.3.5 Red *"feed-forward"*.

Como ya se ha comentado previamente, las componentes *d* y *q* dependen una de la otra. La técnica *"feed-forward"* logra que el ciclo de trabajo tenga las dependencias contrarias que incluirá la etapa de potencia.

Como puede verse en la Figura 4.1, el esquemático para este bloque incluye una variable llamada *decoup* cuyo valor se obtiene de la siguiente forma:

$$decoup = \frac{L \cdot w}{\left(Vdc_{sw} - Vdc_{offset}\right) / \mathrm{Re}\, lVdc} \qquad (4.28)$$

$$V_{dc\_offset} = Vin_{\max} \cdot \frac{1 - \left(Vo_{\max} - Vo_{\min}\right)}{Vin_{\max} - Vin_{\min}} \qquad (4.29)$$

$$\mathrm{Re}\, lVdc = \frac{Vo_{\max} - Vo_{\min}}{Vin_{\max} - Vin_{\min}} \qquad (4.30)$$

Donde $Vo_{\max}$ y $Vo_{\min}$ son la tensión máxima y mínima respectivamente a la salida del conversor analógico/digital, mientras que $Vin_{\max}$ y $Vin_{\min}$ son las tensión máxima y mínima respectivamente a la entrada del conversor analógico-digital.

A continuación se muestra el esquemático correspondiente a este bloque:

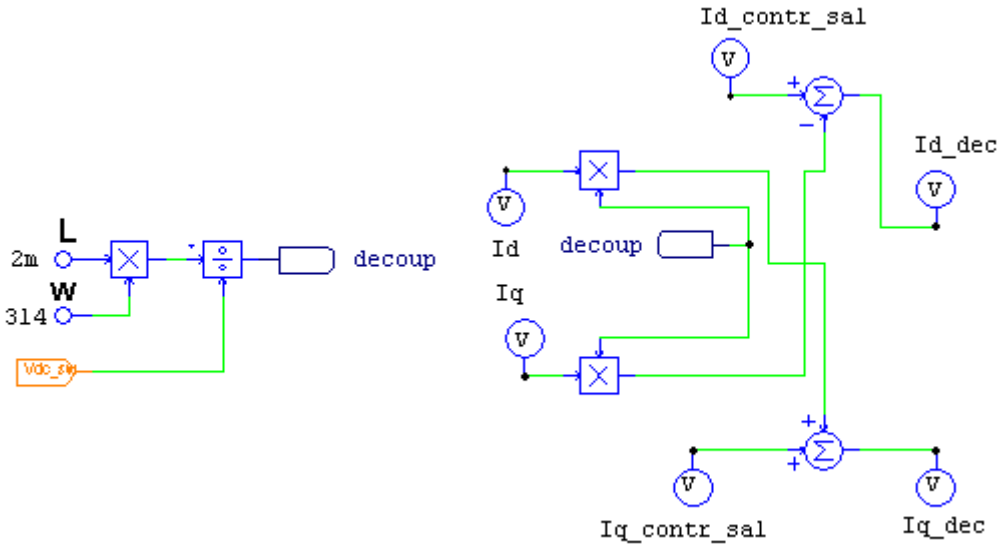


Figura 4.17. Feed-forward.

A la salida de este bloque quedan compensados los efectos de acoplo provocados por las bobinas. Las señales de salida de este sub-bloque son las mostradas en la Figura 4.18, donde es apreciable la mejora con respecto a la salida de los reguladores PI, obtenida previamente a la realimentación mediante *"feed-forward":*

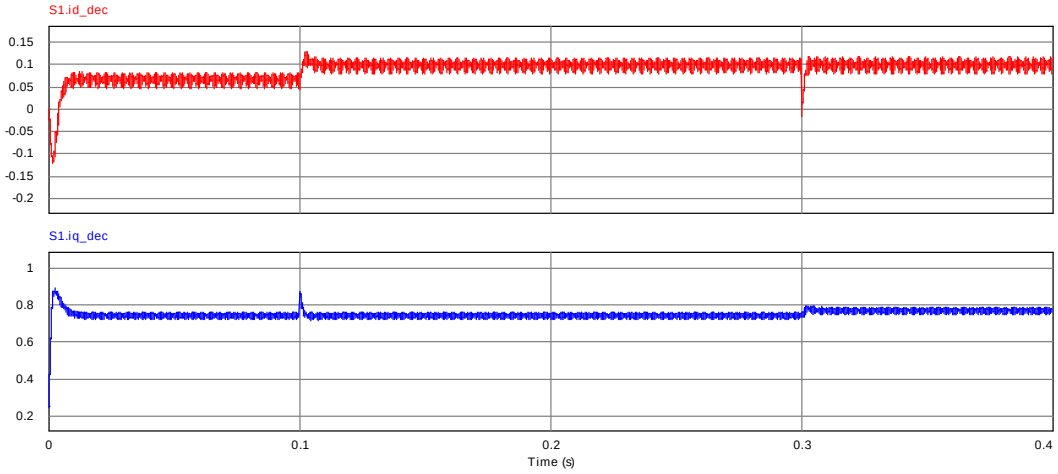


Figura 4.18. Salida del lazo "feed-forward".

4.3.6 Transformada inversa de *Park*

Mediante la transformada inversa de *Park* ya mencionada en la sección 4.5 de este capítulo (Ecuación *4.10*) se deshace el cambio de coordenadas *dq* para tener de nuevo tres tensiones sinusoidales variantes en el tiempo que actuarán como señales de control en la posterior modulación PWM.

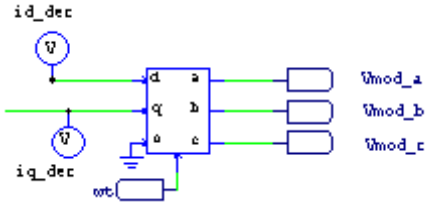


Figura 4.19. Transformada inversa de Park.

A continuación se muestran las señales de salida de este bloque:

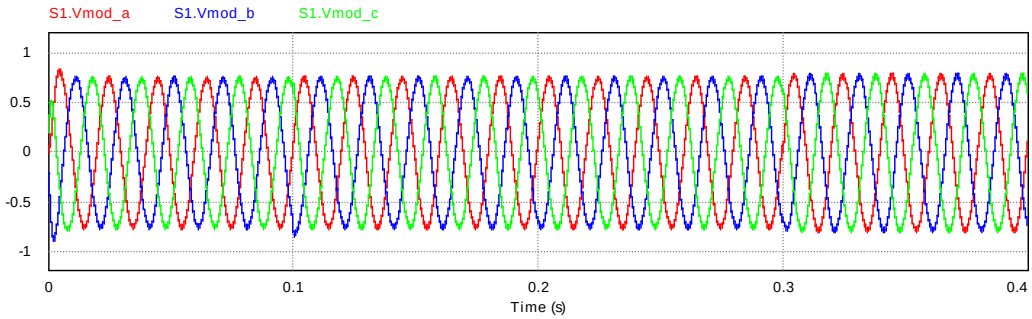


Figura 4.20. Tensión trifásica de control.

4.3.7 Modulación PWM

El objetivo de la modulación de ancho de pulso en un inversor trifásico es modelar y controlar la tensión trifásica de salida en magnitud y frecuencia a partir de una tensión de entrada constante.

Con el propósito de obtener una señal de voltaje a la salida del inversor con la frecuencia deseada, se compara una única onda triangular con las tres tensiones de control senoidales a la frecuencia deseada y desfasadas entre sí 120º.

La frecuencia de la onda triangular corresponde a la frecuencia de interrupción del inversor y por lo general se mantiene constante.

El esquemático correspondiente a la modulación PWM realizada se muestra en la Figura 4.18, donde tal y como puede observarse, las señales sinusoidales de referencia en coordenadas *abc* (V_{mod_a}, V_{mod_b} *y* V_{mod_c}), obtenidas tras la realización del control dq, se comparan con la señal triangular V_{tri} de tensión pico-pico de 2 V y frecuencia 1500 Hz que permite evitar las pérdidas por interrupción dentro del inversor.

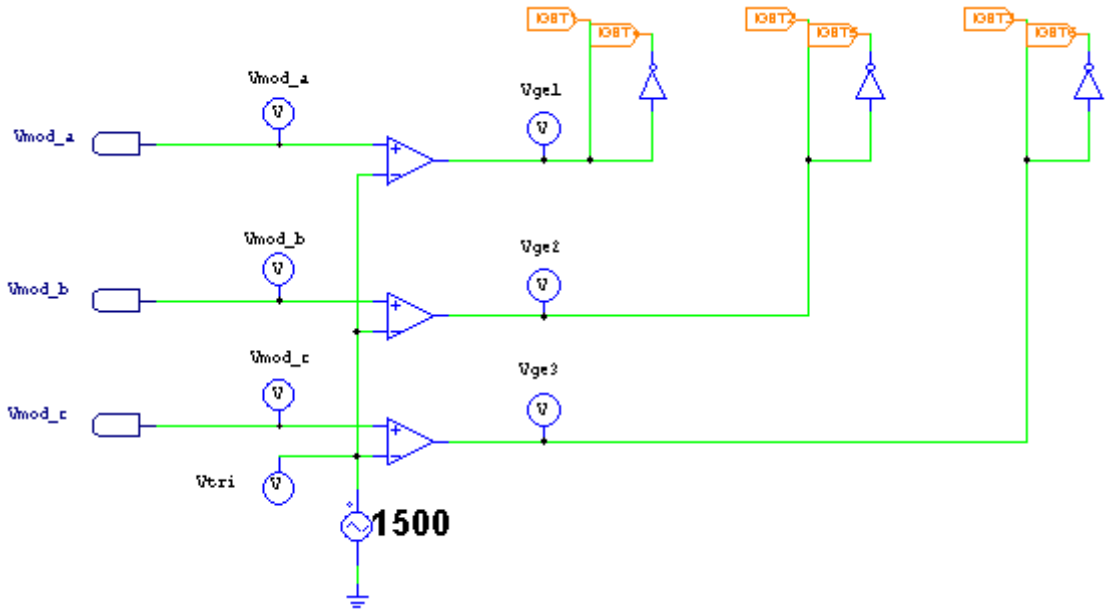


Figura 4.21. Modulador PWM.

En base al resultado de dicha comparación de tensiones, la siguiente tabla muestra el esquema de control en cada una de las patas del inversor trifásico.

	Q1	Q2	Q3	Q4	Q5	Q6
$V_{mod\_a}>V_{tri}$	ON			OFF		
$V_{mod\_a}<V_{tri}$	OFF			ON		
$V_{mod\_b}>V_{tri}$		ON			OFF	
$V_{mod\_b}<V_{tri}$		OFF			ON	
$V_{mod\_c}>V_{tri}$			ON			OFF
$V_{mod\_c}<V_{tri}$			OFF			ON

Tabla 2.1. Modelo de conmutación.

Donde V_{mod} es cada una de las señales sinusoidales desfasadas 120º entre ellas y trabajando a la misma frecuencia y V_{tri} es la señal triangular de modulación.

A continuación se muestran las formas de onda del inversor PWM trifásico.

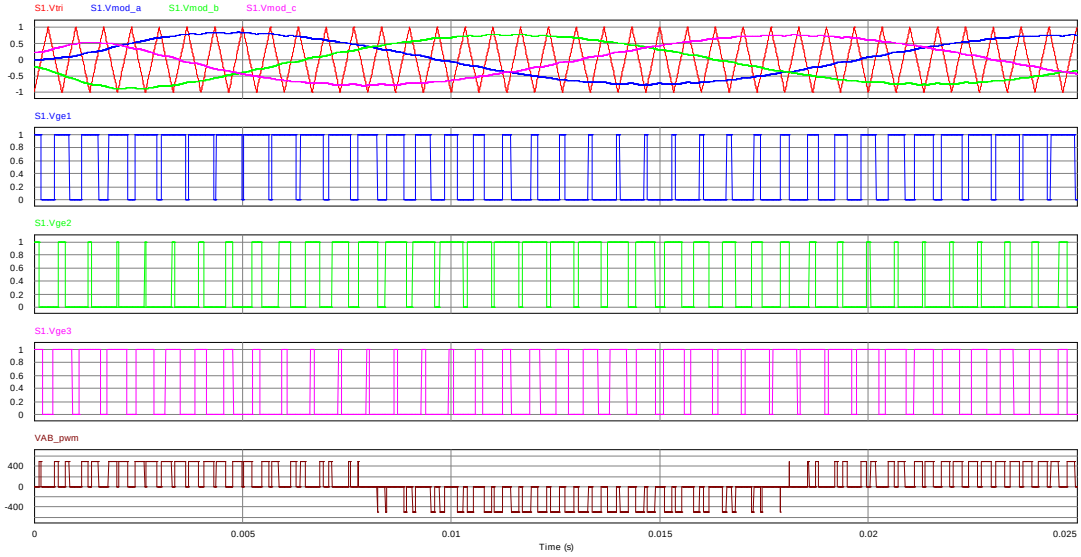


Figura 4.22. Formas de onda del inversor PWM trifásico.

De esta forma, la señal de control se utiliza para activar/desactivar cada uno de los transistores IGBT. Cabe destacar que es inevitable la presencia de armónicos y que existirán por tanto ciertas desviaciones de la señal de onda seno.

Las relaciones entre la señal triangular y la señal de control dependen del valor correspondiente de m_f (ratio de modulación de frecuencias).

Contenido armónico de las tensiones obtenidas.

Interesa analizar el contenido en armónicos de las tensiones de línea. Los armónicos de las tensiones de fase o de salida de cada una de las patas del circuito, son impares si se supone un valor de m_f impar y se encuentran centrados en banda en torno a m_f y sus múltiplos.

Así, una vez obtenido el espectro de armónicos de las tensiones de fase, se intenta reducir ahora el contenido de armónicos de las tensiones de línea. Para ello, considerando las tensiones V_{A_pwm} y V_{B_pwm} del inversor, puesto que estas tensiones se encuentran desfasadas 120º de la frecuencia fundamental, entre los armónicos de dichas señales a frecuencia f_s, es decir, armónicos de orden $n=m_f$, existirá una diferencia de fase de:

$$\left(\phi_{AN}\right)_{mf} - \left(\phi_{BN}\right)_{mf} = (120 \cdot m_f)^{\circ} \qquad (4.38)$$

De esta forma, si se elige m_f impar y múltiplo de tres, esta diferencia de fase se hace nula al ser múltiplo de 360º, es decir, se eliminan los armónicos de orden $n=m_f$ de la tensión de línea V_{AB}. De igual forma, todos los armónicos cuyo orden sea múltiplo impar de m_f también se anulan.

4.4 Control digital.

Cabe destacar que el estudio realizado hasta ahora corresponde a un convertidor de potencia con control analógico, cuyo diagrama de bloques se encuentra plasmado en la Figura 4.1.

A partir de este diseño, se plantea diseñar un control digital, para lo que se precisan conversores analógico-digital que conviertan las señales analógicas de corriente y de tensión en magnitudes discretas, de forma que los conversores A/D realicen una lectura de la magnitud física y no cambien este valor hasta que no se le ordene una nueva lectura.

Los conversores A/D tienen varias fuentes de error que hacen que su función de transferencia difiera del caso ideal. Entre ellas se pueden diferenciar dos grupos de errores: los estáticos y los dinámicos, que por otra parte son errores intrínsecos para cualquier conversión A/D.

Un parámetro a tener en cuenta en los conversores A/D, es la sensibilidad al ruido. Los convertidores conmutados son un medio ruidoso por naturaleza, con grandes derivadas y rizados de corriente.

En la Figura 4.23 se representa la función de transferencia de un conversor A/D ideal.
De aquí se pueden definir las especificaciones estáticas del conversor A/D.

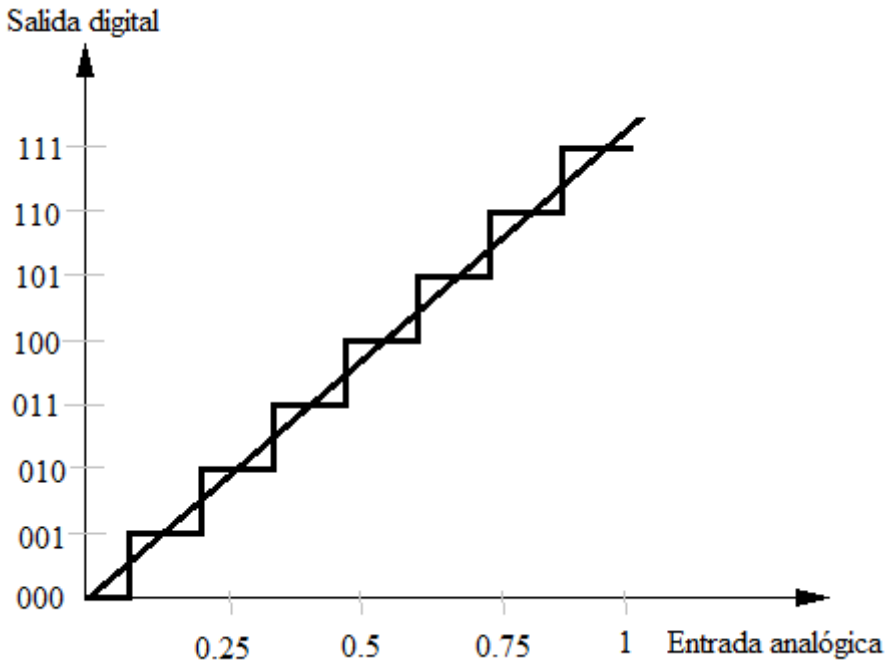


Figura 4.23. Cuantificador ideal.

Hasta ahora se han visto las funciones de transferencia del filtro paso bajo y del regulador PI en el dominio continuo G(s), con lo que es necesario realizar una discretización.

El siguiente código Matlab muestra la obtención de los parámetros discretizados tanto para el filtro como para el regulador:

```
% FILTRO DE ENTRADA
disp('Filtro de entrada');
% Creación de la fdt continua
num = [0 1];
den = [0.0003183 1];
disp('Filtro de entrada: FDT(S)');
fdtFil_s = tf( num, den )
% Discretización
disp('Filtro de entrada: FDT(Z)');
fdtFil_z = c2d( fdtFil_s, Ts, 'tustin' )
```

```
%REGULADOR PI
disp(' ');
disp('Regulador PI');
%Valores de constantes del regulador
Kp = 0.0141733;
Ki = 0.00339366;
%Creación de numerador y denominador
num = [Kp*Ki Kp];
den = [Ki 0];
%Creación de la fdt en continuo
disp('Regulador PI: FDT(S)');
fdtPI_s = tf( num, den )
%Discretización
disp('Regulador PI: FDT(Z)');
fdtPI_z = c2d( fdtPI_s, Ts, 'tustin' )
```

Así pues, al diagrama de bloques mostrado en la Fura 4.1 habría que añadirle los conversores A/D necesarios tal y como se muestra.

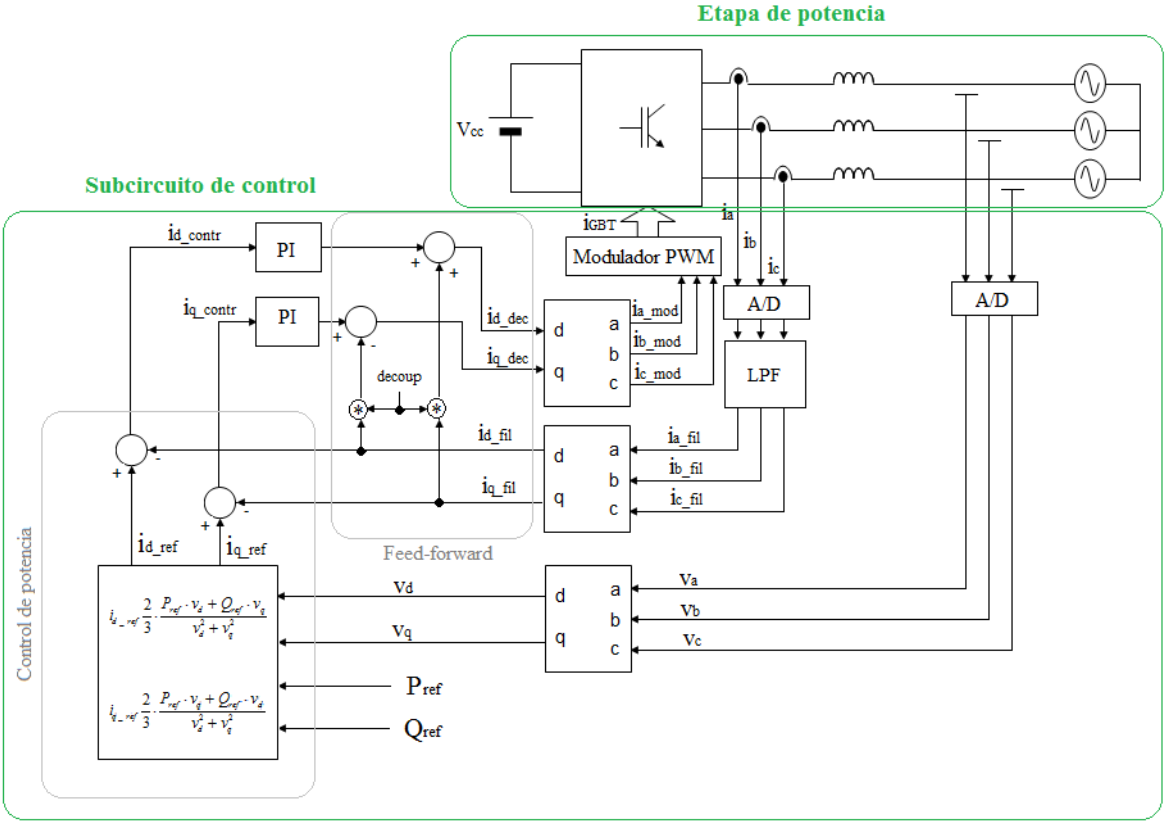


Figura 4.24. Diagrama de bloques del sistema inversor conectado a red con control digital.

Cabe destacar que debido a la discretización en los bloques Filtro Paso bajo y Regulador PI, para el modelo de control digital se emplearán los nuevos coeficientes calculados mediante Matlab para el filtro y el regulador En el próximo capítulo, ya que la implementación en VHDL se ha realizado para el modelo digital, se detallarán los nuevos valores de los coeficientes en cada bloque. Además, como también se verá en el próximo capítulo, serán necesarios añadir algunos reescalados a la salida de los filtros de entrada y a la entrada de los reguladores PI.

Capítulo 5

Diseño digital del control del sistema

A lo largo de este capítulo, se procede a realizar una descripción acerca de cómo se ha llevado a cabo el diseño y validación de una solución propuesta para el control digital dq de un inversor conectado a red. Para ello, se mostrarán uno a uno, los bloques en que se ha dividido el circuito descrito en el apartado anterior para su posterior diseño digital. Así mismo, se mostrará el correcto funcionamiento de cada uno de los bloques implementados mediante la comparación de los resultados obtenidos mediante la simulación en Modelsim y PSIM.

5.1 Subcircuito de control digital

Antes de comenzar con la descripción, cabe destacar que se ha realizado un modelo simulable de todos los bloques mencionados en el capítulo anterior. La principal razón para ello es poder demostrar de forma completa la eficacia de una metodología de diseño mixta, analógico-digital. El diseño digital de los bloques, su simulación mediante una herramienta CAD comercial de diseño digital (Modelsim), su integración en un sistema analógico (el inversor descrito en el capítulo anterior) y la cosimulación analógico-dogital de todo el sistema mediante la conexión entre una herramienta CAD comercial de diseño analógico (PSIM) y la anteriormente mencionada Modelsim.

La implementación *hardware* de los bloques del control digital escapa del objetivo principal de este Proyecto Final de Carrera, si bien bien se ha considerado en las etapas de diseño de la arquitectura y diseño detallado de los bloques. La presencia de operaciones matemáticas complejas, tales como funciones trigonométricas, productos, divisiones, multiplicaciones matriciales, etc. hace que la implementación hardware de cabida a futuros trabajos y estudios.

Para el modelo simulable del circuito de control, se han definido todos los objetos del diseño, señales, constantes y variables como reales o de punto flotante, que permiten representar números enteros y racionales en un rango entre -1e38 a 1e38, para lograr un mayor acercamiento entre los valores finales obtenidos mediante la simulación de la descripción VHDL y los esperados mediante PSIM.

De igual forma que se hizo en el capítulo anterior, se detallan los diferentes bloques a implementar mediante VHDL sobre el esquemático PSIM del modelo de control digital, cuyo diagrama de bloques se ha expuesto previamente en la Figura 4.24.

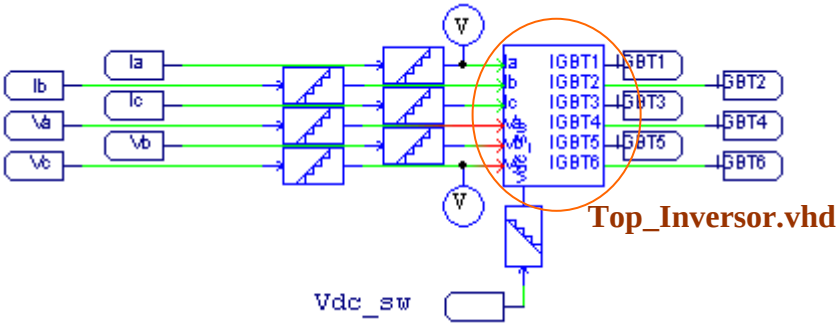


Figura 5.1. Subcircuito de control digital.

Uno de los convertidores analógico/digital más comunes es el de 12 bits, como es, por ejemplo, el AD7887 de *"Analog Devices"* que se ha usado como referencia. Dicho ADC admite una frecuencia de muestreo de 125kHz y su rango de entrada comprenderá desde -56V a 56V, mientras que el rango de salida estará comprendido entre 0 y 5V.

A continuación se muestran los diferentes sub-bloques del sistema de control considerados en el capítulo anterior y su correspondencia con cada uno de los bloques implementados en VHDL. Cabe destacar que el modulador PWM se incluirá en el fichero final ***Top_Inversor.vhd*** que engloba todos los sub-bloques observados en la Figura *5.2.*

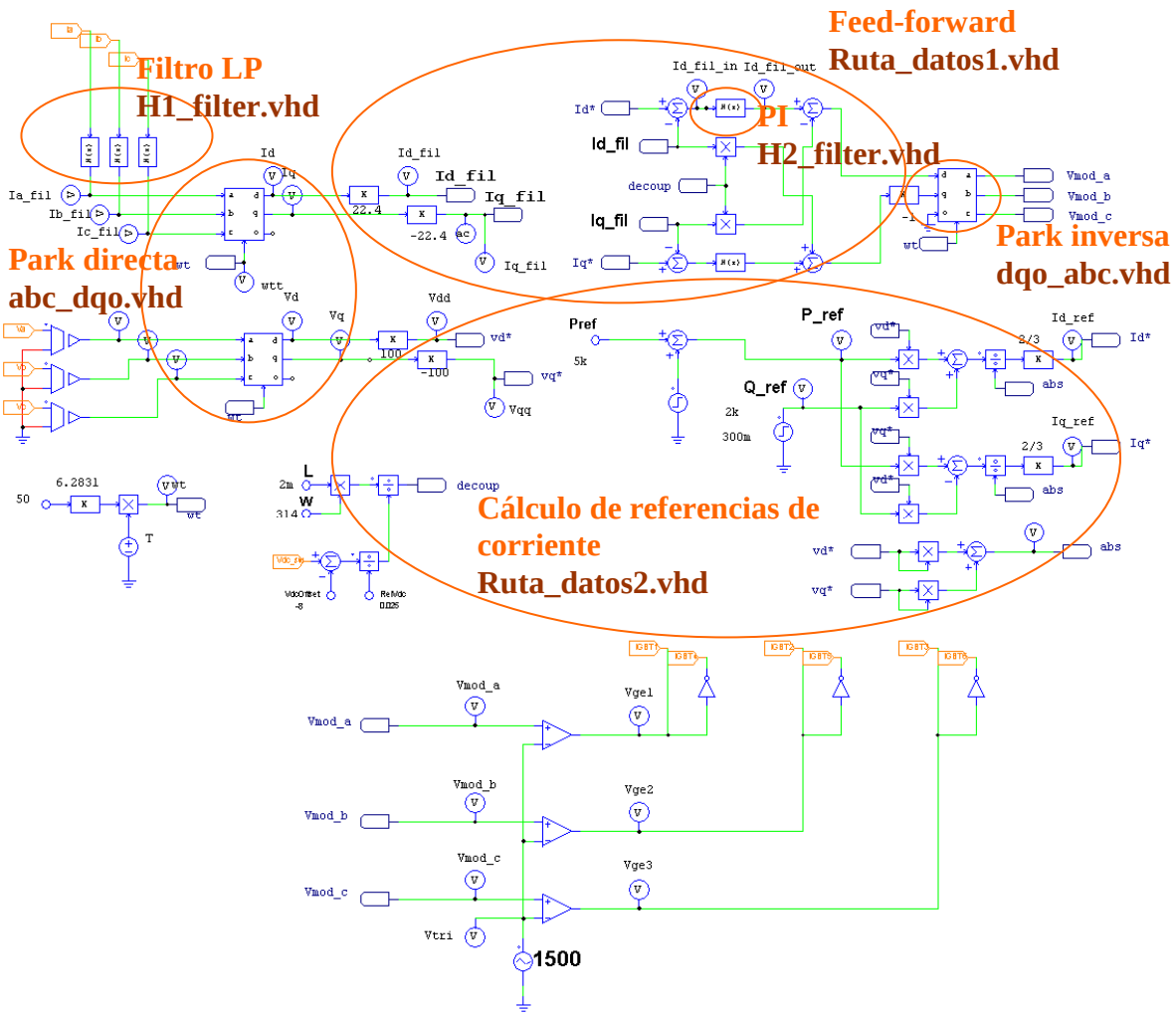


Figura 5.2.Relación bloques del subcircuito de control digital y ficheros .vhd.

5.2 H1_filter

5.2.1 Descripción

Una vez tomada la decisión de realimentar las componentes de corriente filtradas, para no estar limitados a una elevada frecuencia de conmutación que reduzca el rizado, en primer lugar se ha llevado a cabo la implementación del filtro denominado H1 a fin de lograr tal efecto.
Este filtro será utilizado para cada una de las tres corrientes sinusoidales de la red I_a, I_b e I_c.

La entidad para el filtro de entrada, se muestra en la siguiente figura.

Figura 5.3. Entidad del filtro paso bajo de entrada.

Entradas:

- *clk*: reloj de sincronismo del circuito para todos los biestables del mismo.
- *reset*: señal de inicialización asíncrona para todos los biestables del circuito.
- $h1_{in}$: valor digitalizado de la corriente de red.

Salidas:

- $h1_{out}$: valor digital de la corriente de red filtrada.

Como ya se ha mencionado, dicho filtro se aplicará sobre cada una de las señales de red I_a, I_b e I_c, tal y como se observa en la Figura 5.2, por lo que será necesario mapear tres veces este filtro en el fichero final ***Top_Inversor.vhd***, como se verá en el próximo capítulo.

Puesto que se han aplicado conversores A/D a las señales de entrada a este filtro, los coeficientes del filtro de entrada para control digital, difieren de los indicados en el capítulo anterior para el caso analógico, y se han obtenido mediante el código matlab mostrado en la sección 4.4.

El filtro de primer orden a implementar a una frecuencia de muestreo de 125 kHz, tiene la siguiente función de transferencia en el dominio Z:

$$H(z) = \frac{b_0 \cdot z^N + b_1 \cdot z^{N-1}}{a_0 \cdot z^N + a_1 \cdot z^{N-1}} \qquad (5.1)$$

Los coeficientes obtenidos para el control digital son:

a_0=1
a_1=-0.9752
b_0=0.01242
b_1=0.01242

Al ser $a_0 = 1$, la función de transferencia de este filtro puede ser expresada mediante la siguiente ecuación en diferencias:

$$a_0\, h1_{out}[n] + a_1\, h1_{out}[n-1] = b_0\, h1_{in}[n] + b_1\, h1_{in}[n-1] \qquad (5.2)$$

Como condición inicial, se supondrá $h1_{in}[n]=0$.

5.2.2 Ruta de datos

En la siguiente figura se muestra la ruta de datos para el diseño digital.

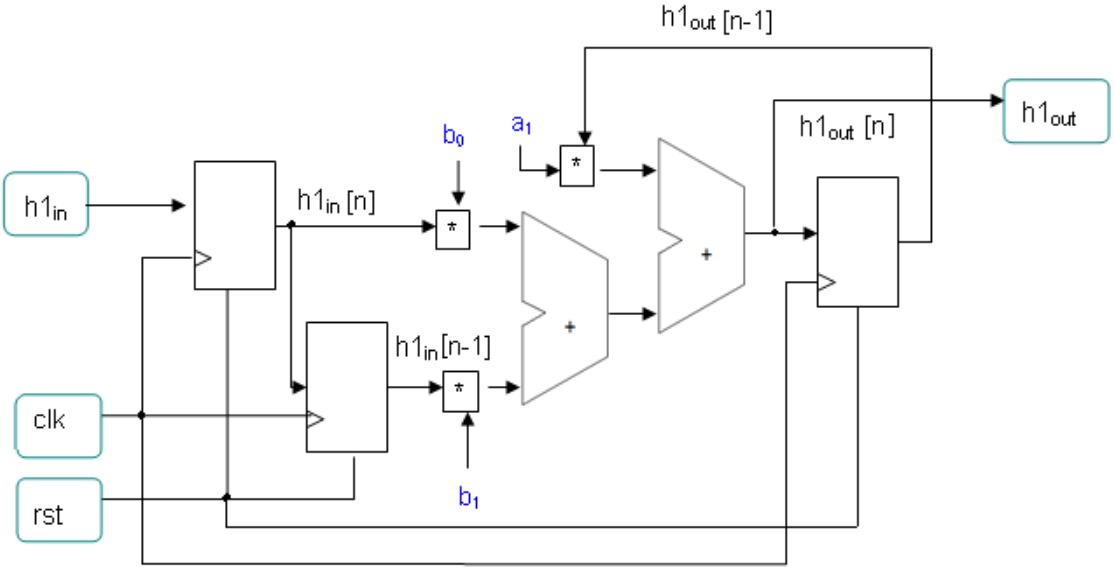


Figura 5.4. Ruta de datos del filtro paso bajo de entrada.

5.2.3 Descripción en VHDL

Bibliotecas:

Se indican en primer lugar las bibliotecas necesarias, que contienen los tipos de datos necesarios:

```
--Para usar el tipo
use ieee.std_logic_1164.all;
--Para usar funciones aritméticas de finidas (suma, resta, producto)
use ieee.std_logic_arith.all;
```

Entidad:
Seguidamente se procede a la declaración de la entidad ***H1_filter***, donde se listan los genéricos reales que serán los coeficientes ya descritos para el filtro, así como los puertos de entrada y salida.
Las entradas *clk* y *reset* son de tipo "*std_logic*", mientras que la entrada *h1_in* y la salida *h1_out* serán de tipo real.

Arquitectura:
La arquitectura H1 de la entidad ***H1_filter*** consta de cuatro registros que almacenarán los valores de las entradas y las salidas en el instante actual y en el instante previo.

En la arquitectura se describen los cuatro registros que, en cada ciclo de reloj irán almacenando los valores de entrada y salida del filtro en el instante actual y en el previo.
Además, de forma combinacional se calcula la ecuación en diferencias expresada en la Ecuación 5.2.

5.2.4 Verificación de resultados

Con el fin de poder llevar a cabo la simulación en Modelsim, se ha implementado un banco de pruebas donde se mapea la entidad ***H1_filter*** y se declaran tres procesos: uno para la generación de la señal de reset, otro para la señal de reloj y un último proceso que permite una lectura en cada ciclo de reloj de las señales de entrada. Dichas señales de entrada se encuentran en un fichero guardado en el mismo directorio del proyecto y ha sido obtenido previamente de la simulación mediante PSIM. La lectura de estos datos de entrada se producirá hasta el final del fichero. Además, puesto que el fichero obtenido de PSIM proporciona una muestra cada 5 µs, se fija el periodo de la señal *clk* también a 5µs para que tenga lugar una lectura del fichero en dicho intervalo temporal.

A continuación se muestra la comparativa de resultados obtenidos mediante PSIM y Modelsim a la salida de este filtro, usando como señal de entrada I_a, ya que si el resultado obtenido es correcto, lo será también aplicando el filtro sobre el resto de entradas.

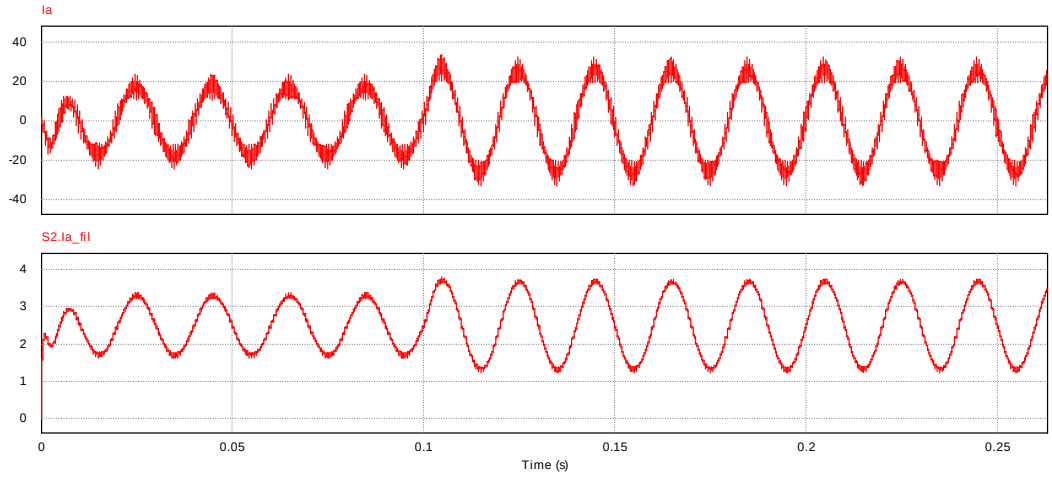


Figura 5.5. Entrada y salida simuladas mediante PSIM para el Filtro Paso Bajo.

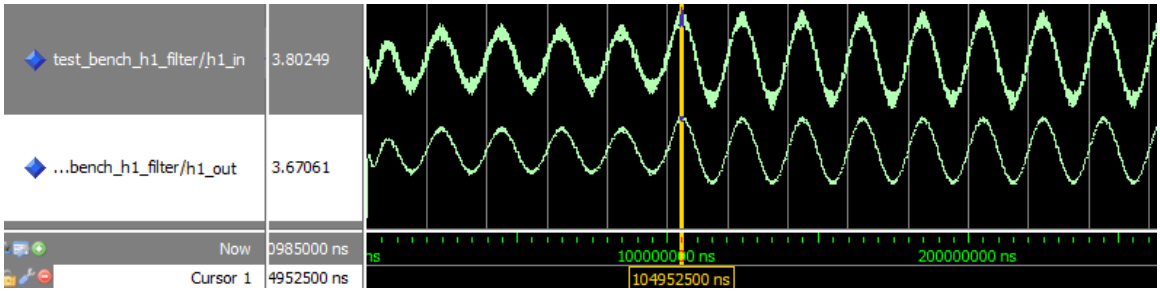


Figura 5.6.Entrada y salida simuladas mediante Modelsim para el Filtro Paso Bajo.

Comprobando los resultados obtenidos mediante Modelsim con el fichero de salida de dicho filtro para PSIM, se pudo comprobar que efectivamente se obtienen los valores esperados. Será posible pasar a la implementación del siguiente bloque del sistema con la garantía de que este bloque funciona como debe, ya que se ha obtenido un error relativo

más que aceptable ($\bar{\varepsilon}$=0.86 %), promediando los errores relativos para las primeras cincuenta muestras tal como se muestra a continuación:

$$\left(\varepsilon = \frac{out_{PSIM} - out_{Modelsim}}{out_{PSIM}} \right) \tag{5.3}$$

$$\bar{\varepsilon} = \frac{\sum_{N} \varepsilon}{N} \tag{5.4}$$

5.3 abc_dqo

5.3.1 Descripción

En este bloque se realiza el diseño digital de la transformación directa de *Park*, que tal como se comentó en el capítulo previo de este proyecto, viene expresada por la siguiente ecuación:

$$T_{abc/dqo} = \frac{2}{3} \cdot \begin{bmatrix} \cos(w \cdot t) & \cos\left(w \cdot t - \frac{2\pi}{3}\right) & \cos\left(w \cdot t + \frac{2\pi}{3}\right) \\ -sen(w \cdot t) & -sen\left(w \cdot t - \frac{2\pi}{3}\right) & -sen\left(w \cdot t + \frac{2\pi}{3}\right) \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \end{bmatrix} \tag{5.5}$$

De manera que las ecuaciones resultantes son:

$$d = \frac{2}{3}\left[\cos(w \cdot t) \cdot a + \cos\left(w \cdot t - \frac{2 \cdot \pi}{3}\right) \cdot b + \cos\left(w \cdot t + \frac{2 \cdot \pi}{3}\right) \cdot c \right] \tag{5.6}$$

$$q = \frac{2}{3}\left[\sin(w \cdot t) \cdot a + \sin\left(w \cdot t - \frac{2 \cdot \pi}{3}\right) \cdot b + \sin\left(w \cdot t + \frac{2 \cdot \pi}{3}\right) \cdot c \right] \tag{5.7}$$

Puesto que tal y como se ha comentado en la introducción a este capítulo, para este proyecto se ha realizado un modelo simulable del circuito digital, no es necesario realizar ninguna aproximación algorítmica que proporcione un resultado aproximado de las funciones seno y coseno. No obstante, en trabajos futuros que consideren una descripción sintetizable, sería conveniente hacer uso de alguna de las aproximaciones de que se disponen.

Así, haciendo uso de la aproximación de Taylor, mediante sencillas operaciones de sumas y productos, podemos llegar a valores muy válidos a efectos prácticos.

$$f(x) = f(a) + \frac{f_a^{'}}{1!}(x-a)^1 + \frac{f_a^{''}}{2!}(x-a)^2 + ... + \frac{f_a^{n}}{n!}(x-a)^n \tag{5.8}$$

Función seno:

$$f(x) = sen(x) \rightarrow f(0) = 0 \tag{5.9}$$

$$f^{'}(x) = \cos(x) \rightarrow f^{'}(0) = 1 \tag{5.10}$$

$$f''(x) = -sen(x) \rightarrow f''(0) = 0 \quad (5.11)$$

$$f'''(x) = -\cos(x) \rightarrow f'''(0) = -1 \quad (5.12)$$

$$f^{v}(x) = sen(x) \rightarrow f^{v}(0) = 0 \quad (5.13)$$

Por lo tanto, APROXIMACIÓN 1:

$$sen(x) \cong x - \frac{x^3}{3!} \quad (5.14)$$

APROXIMACIÓN 2:

$$sen(x) \cong x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} \quad (5.15)$$

Función coseno:

$$f(x) = \cos(x) \rightarrow f(0) = 1 \quad (5.16)$$

$$f'(x) = -sen(x) \rightarrow f'(0) = 0 \quad (5.17)$$

$$f''(x) = -\cos(x) \rightarrow f''(0) = -1 \quad (5.18)$$

$$f'''(x) = sen(x) \rightarrow f'''(0) = -0 \quad (5.19)$$

$$f^{v}(x) = \cos(x) \rightarrow f^{v}(0) = 1 \quad (5.20)$$

Por lo tanto, APROXIMACIÓN 1:

$$\cos(x) \cong 1 - \frac{x^2}{2!} \quad (5.21)$$

APROXIMACIÓN 2:

$$\cos(x) \cong 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} \quad (5.22)$$

Para el cálculo de estas potencias y sumas, se puede realizar una arquitectura totalmente desplegada (área máxima y tiempo de cálculo mínimo) o una arquitectura realimentada, tipo microprocesador, (area mínima y tiempo de cálculo máximo). Dependiendo de los requisitos impuestos al circuito, será deseable elegir aquella aproximación que consiga una solución de compromiso entre la aproximación al valor real y el tiempo de computación que se requiere para llegar a dicho valor aproximado.

Durante la primera fase del Proyecto, se realizó un análisis de la viabilidad de realizar una implementación sintetizable. Pese a que la presencia de divisores en los bloques siguientes, dificultó la tarea, en el apartado de Anexos (abc_dqo.vhd sintetizable) se adjunta el código implementado para el caso de la aproximación de las funciones seno y coseno mediante aproximaciones de Taylor:

La entidad para la transformada de *Park*, será la siguiente:

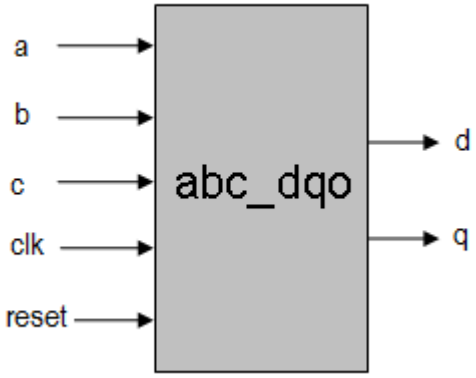


Figura 5.7. Entidad de la transformada de Park directa.

Entradas:

- *clk:* reloj de sincronismo del circuito para todos los biestables del mismo.
- *reset*: señal de inicialización asíncrona para todos los biestables del circuito.
- *a ,b ,c:* señales sinusoidales de entrada.

Salidas:

- *d, q*: señales constantes tras el cambio de sistema de referencia.

5.3.2 Ruta de datos

En el caso del ***abc_dqo*** (así como para el del ***dqo_abc***), se muestra la siguiente ruta de datos:

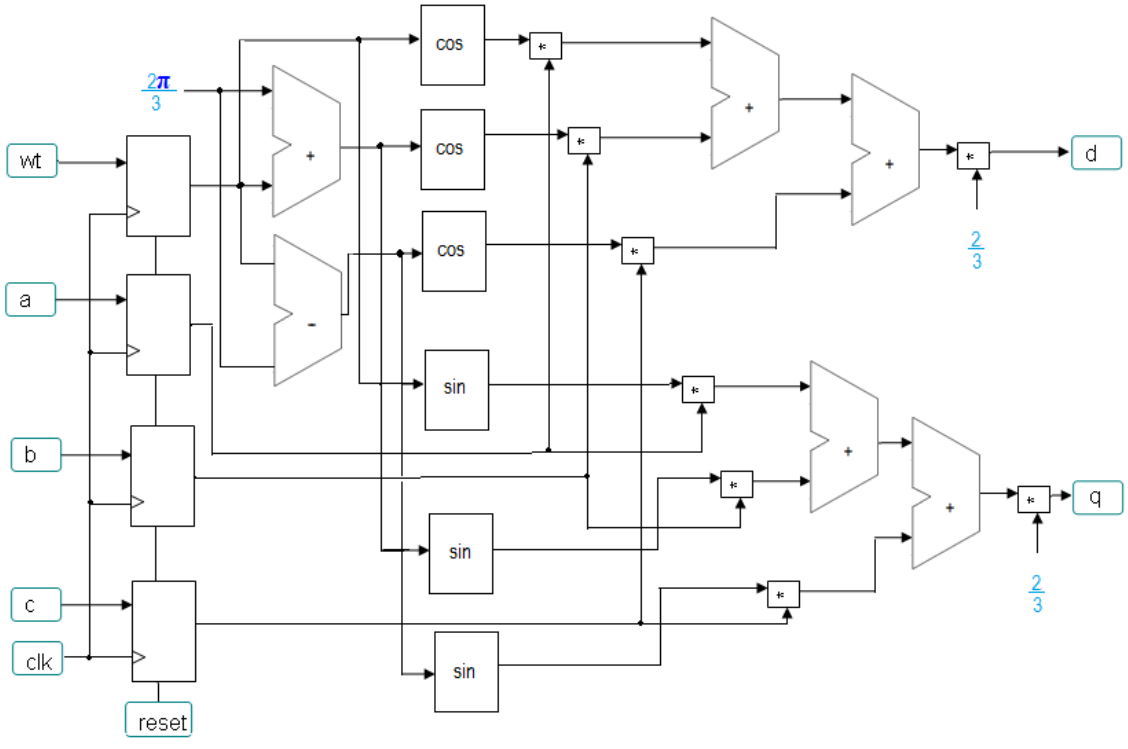


Figura 5.8. Ruta de datos simplificada para la transformada directa de Park.

5.3.3 Descripción en VHDL

Bibliotecas:
Al igual que para el primer bloque ya visto, se procede en primer lugar a la inclusión de las bibliotecas tanto para usar las funciones aritméticas necesarias, como para las constantes reales, necesario en este caso para usar el valor de π.
Entidad:
Posteriormente se declara la entidad abc_dqo, donde se declara un genérico real que será el coeficiente 2/3 de las ecuaciones para d y q mostradas previamente, así como los puertos de entrada y salida. Las entradas clk y reset son de tipo std_logic, mientras que las entradas a, b, c y las salidas d,q serán de tipo real.
Arquitectura:
La arquitectura *a* de la entidad ***abc_dqo*** consta de cuatro señales de entrada que almacenarán los valores de las entradas (*a, b, c* y *wt*) necesarias para obtener cada una de las salidas.

En la arquitectura se describen los cuatro registros que, en cada ciclo de reloj irán almacenando los valores de entrada y salida de la transformada de *Park.*
De forma combinacional se calculan las Ecuacione 5.6 y 5.7.

5.3.4 Verificación de resultados

Nuevamente, con el fin de poder llevar a cabo la simulación en Modelsim, se ha implementado un banco de pruebas donde se mapea la entidad abc_dqo y se declaran tres procesos: uno para la generación de la señal de reset, otro para la señal de reloj y un último proceso que permite una lectura en cada ciclo de reloj de las señales de entrada a, b, c que se encuentran en los respectivos ficheros de entrada. Siguiendo la misma dinámica que en el caso del filtro, dichos ficheros de entrada están guardados en el directorio donde se encuentra el Proyecto implementado, y obtenidos una vez más de la previa simulación mediante PSIM.

Este sub-bloque se aplicará tanto para las corrientes de red filtradas como para las tensiones de red.
Para la comprobación del correcto funcionamiento de este bloque se han analizado los resultados sobre las corrientes filtradas, ya que si para este caso se obtienen los valores esperados, lo mismo ocurrirá al aplicar este bloque sobre las tensiones.

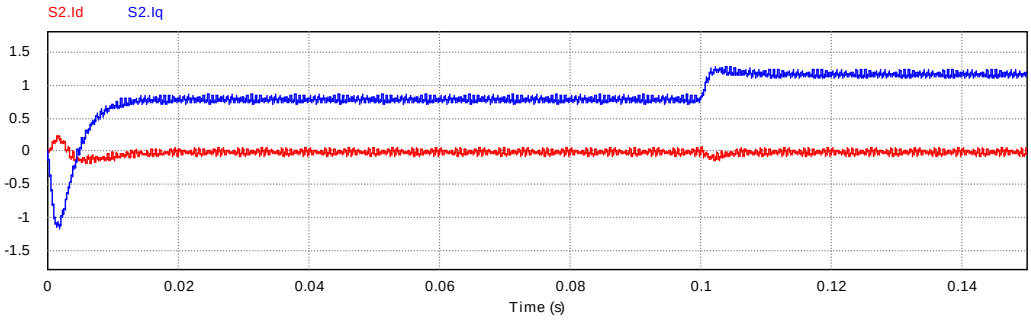


Figura 5.9. Salidas d y q obtenidas mediante PSIM.

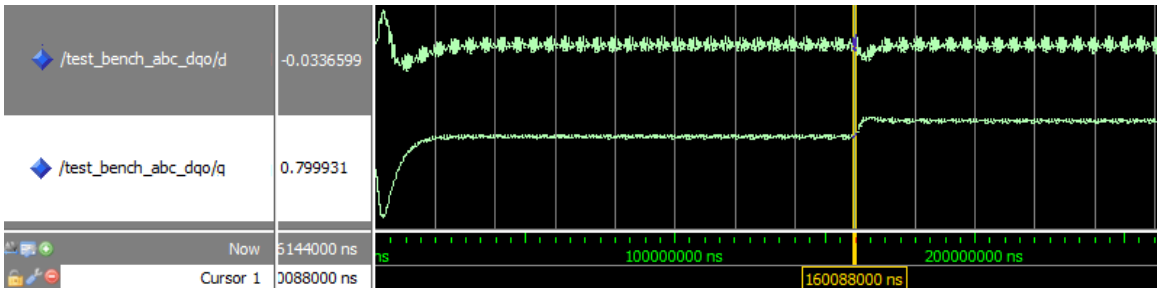


Figura 5.10. Salidas d y q obtenidas mediante Modelsim.

Tras realizar la simulación en Modelsim sobre las tres entradas Ia_fil, Ib_fil e Ic_fil, puede comprobarse que efectivamente los resultados obtenidos con el banco de pruebas de Modelsim se asemejan bastante a los resultados obtenidos mediante PSIM, ya que procediendo de manera idéntica que en caso de ***H1_filter***, aplicando las Ecuaciones 5.3 y 5.4, se han obtenido errores relativos nuevamente aceptables ($\bar{\varepsilon}$=2.03% para la salida *d* y $\bar{\varepsilon}$=1.9 % para la salida *q*), habiendo promediando los errores relativos para las primeras cincuenta muestras.

5.4 Ruta_Datos1

5.4.1 Descripción

En este tercer sub-bloque denominado ***Ruta_Datos1***, se implementará la generación de las referencias I_{d_ref} e I_{q_ref} y la compensación de potencias activa y reactiva.

Como se puede ver en la Figura 5.11, este sub-bloque tomará como entradas las respectivas salidas de cada uno de los bloques ***abc_dqo***, generando a su salida las señales que serán necesarias para el sub-bloque que trata el desacoplo de las componentes *d,q* mediante la red *"feed-forward"*.

Entradas:

- *clk:* reloj de sincronismo del circuito para todos los biestables del mismo.
- *reset*: señal de inicialización asíncrona para todos los biestables del circuito.
- *Pref, Qref*: potencias activa y reactiva.
- *Id, Iq, Vd, Vq*: corrientes y tensiones de red en sistema de referencia *dq*.

Salidas:

- *Id_sal, Iq_sal:* Corrientes de línea una vez compensados los efectos de potencia activa y reactiva.
- *Id_fil, Iq_fil*: Estas salidas serán las corrientes filtradas y transformadas al dominio *dq* pero además aplicando el escalado necesario debido a la digitalización de las señales de entrada al sistema de control, que provoca un cambio en los coeficientes del filtro de entrada. Así pues, será necesario aplicar el siguiente escalado sobre las señales de salida del conversor ***abc_dqo***:

$$escalado = \frac{Vin_{\max} - Vin_{\min}}{Vo_{\max} - Vo_{\min}} \quad (5.23)$$

Donde $Vo_{\max}$ y $Vo_{\min}$ son la tensión máxima y mínima respectivamente a la salida del conversor analógico/digital, mientras que $Vin_{\max}$ y $Vin_{\min}$ son las tensión máxima y mínima respectivamente a la entrada del conversor analógico/digital.

En la siguiente figura se muestra la entidad correspondiente a este bloque:

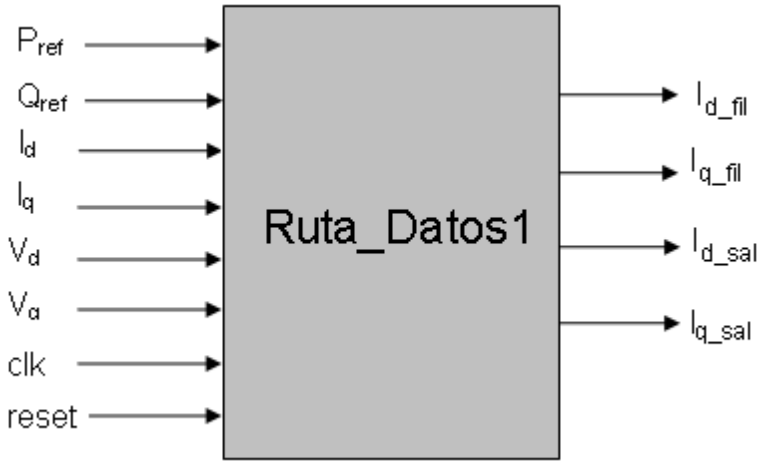


Figura 5.11. Entidad de Ruta_Datos1.

5.4.2 Ruta de datos

En la Figura 5.12 se muestra la ruta de datos del bloque que se describirá con el lenguaje VHDL.

Las constantes K_1 y K_2 son el escalado calculado mediante la Ecuación 5.23 y que aplica a las corrientes filtradas i_d e i_q.

K_1=22.4
K_2=-22.4

Las constantes K_3 y K_4 son el escalado aplicado esta vez sobre las tensiones v_d y v_q y que se calcula también a partir de la Ecuación 5.23.

K_3=100
K_4=-100

La constante K_5 sin embargo, corresponde al término 2/3 de las siguientes ecuaciones ya explicadas en el capítulo anterior:

$$i_{d\_ref} = \frac{2}{3} \cdot \frac{P_{ref} \cdot v_d + Q_{ref} \cdot v_q}{v_d^2 + v_q^2} \quad (5.24)$$

$$i_{q\_ref} = \frac{2}{3} \cdot \frac{P_{ref} \cdot v_q + Q_{ref} \cdot v_d}{v_d^2 + v_q^2} \quad (5.25)$$

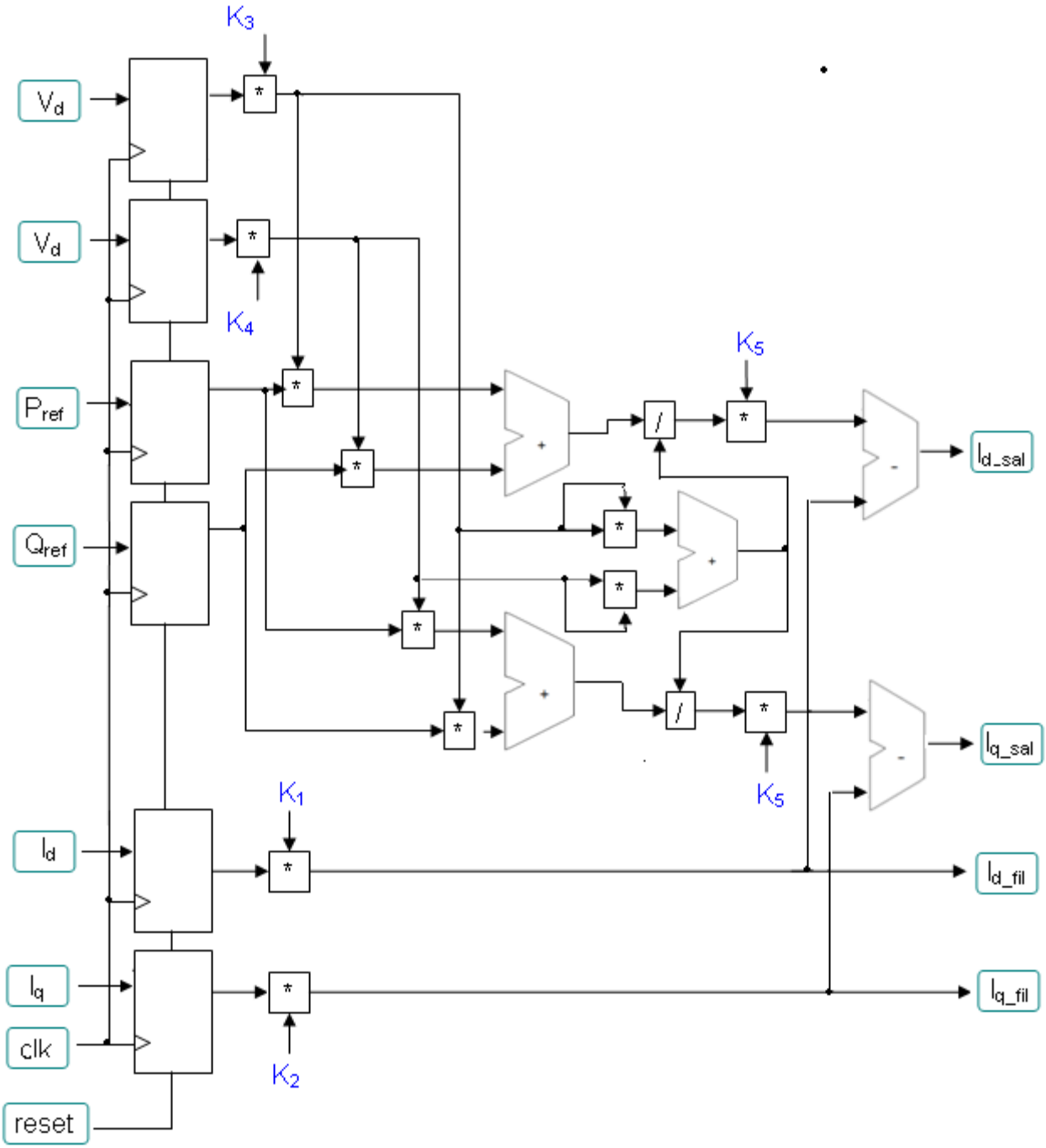


Figura 5.12. Ruta de datos para Ruta_Datos1.

5.4.3 Descripción en VHDL

Bibliotecas:
Una vez más se incluyen las bibliotecas necesarias para usar las funciones aritméticas.
Entidad:
Se declara la entidad ***Ruta_Datos1***, donde se incluyen los genéricos reales correspondientes a las constantes descritas previamente.

Las entradas de esta entidad son *clk* y *reset*, de tipo "*std_logic*", y *Pref, Qref,* I_d, I_q, V_d, V_q de tipo real. Las salidas, también de tipo real, son I_{d_sal}, I_{q_sal}, I_{d_fil} e I_{q_fil}.

Arquitectura:

La arquitectura Ruta1 de la entidad ***Ruta_Datos1*** consta esta vez de seis señales correspondientes a las entradas registradas, para poder realizar operaciones intermedias con los valores que tomen dichas entradas.

En la arquitectura se describen los registros que, en cada ciclo de reloj irán almacenando los valores de entrada y salida de este bloque.

Nuevamente, de forma combinacional se calculan las Ecuacione 5.24 y 5.25.

5.4.4 Verificación de resultados

Siguiendo la misma metodología que en los sub-bloques previos, se ha descrito un banco de pruebas donde se mapea la entidad Ruta_Datos1 y se declaran tres procesos: uno para la generación de la señal de reset, otro para la señal de reloj con periodo 5µs y un último proceso que permite una lectura en cada ciclo de reloj de las señales de entrada que se encuentran en los ficheros guardados en el mismo directorio que el proyecto implementado, y que se han obtenido una vez más de la simulación mediante PSIM.

A continuación se muestran las salidas del convertidor abc_dqo una vez aplicado el escalado indicado en la Ecuación 5.23, es decir, I_{d_fil} e I_{q_fil}.

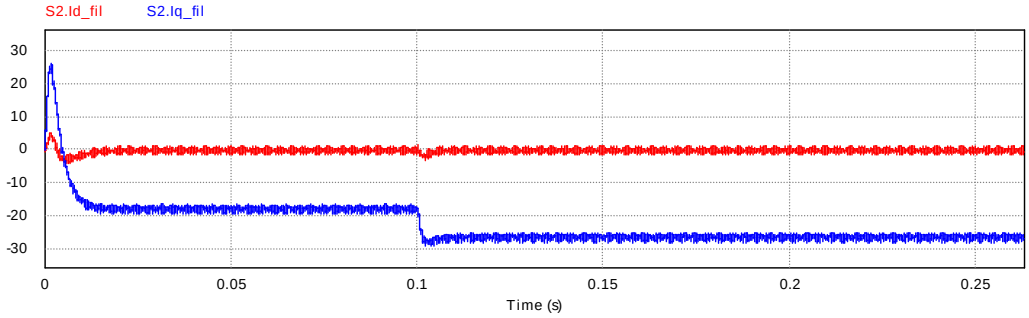


Figura 5.13. Salidas I_{d_fil} e I_{q_fil} obtenidas mediante PSIM.

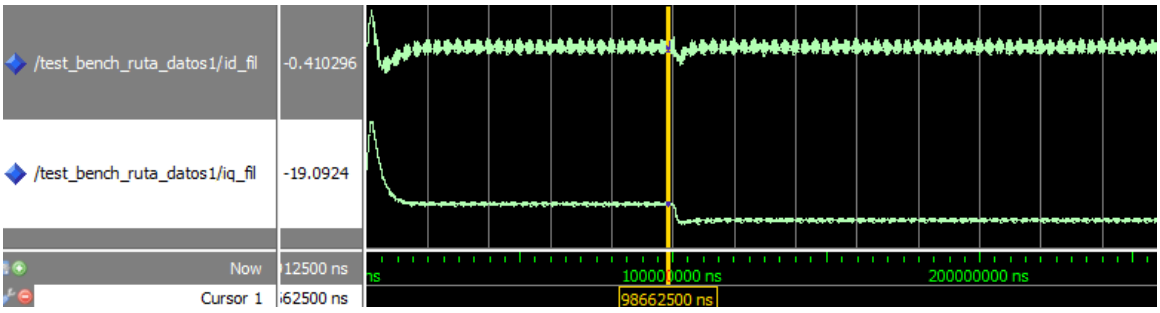


Figura 5.14. Salidas I_{d_fil} e I_{q_fil} obtenidas mediante Modelsim.

En la siguiente figura se muestran las otras dos salidas de este bloque: I_{d_sal}, I_{q_sal} simuladas con PSIM primero y mediante Modelsim después.

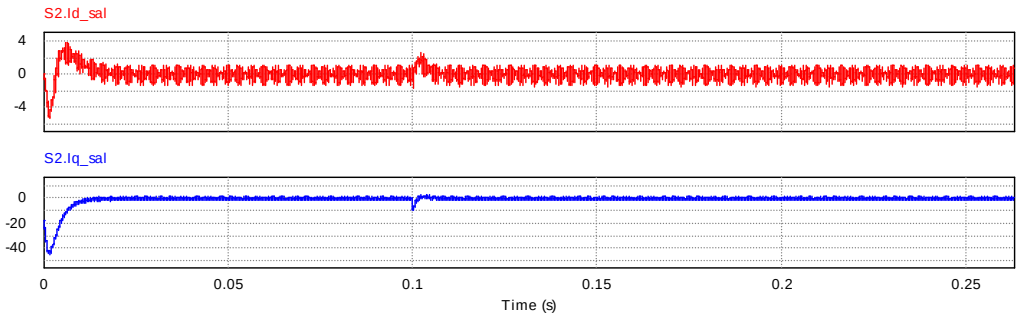


Figura 5.15. Salidas I_{d_sal} e I_{q_sal} obtenidas mediante PSIM.

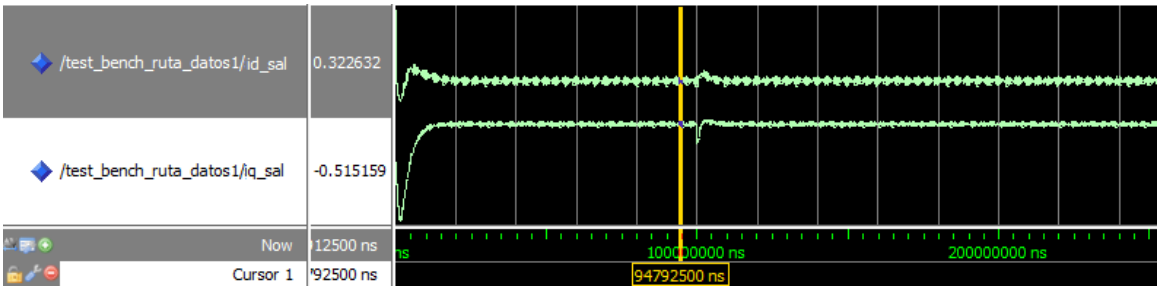


Figura 5.16. Salidas I_{d_sal} e I_{q_sal} obtenidas mediante Modelsim.

Una vez más, se procede de manera idéntica que los sub-bloques previos, aplicando las Ecuaciones 5.3 y 5.4 sobre cada una de las salidas de este bloque, y obteniendo los siguientes errores relativos promediados para las primeras cincuenta muestras: $\bar{\varepsilon}$=2.13% para I_{d_sal}, $\bar{\varepsilon}$=1.83% para I_{q_sal}, $\bar{\varepsilon}$=2.05 para I_{d_fil} y $\bar{\varepsilon}$=1.94% para I_{q_fil}. Todos ellos resultados razonables. Así pues, se puede pasar al siguiente bloque con la certeza de que este bloque funciona como es debido.

5.5 H2_filter

5.5.1 Descripción

En ***H2_filter*** se lleva a cabo la imlementación VHDL del regulador PI. En la siguiente figura se muestra la entidad para dicho regulador:

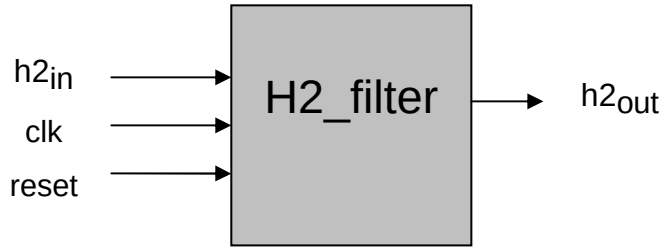


Figura 5.17. Entidad del regulador.

Entradas:

- *clk*: reloj de sincronismo del circuito para todos los biestables del mismo
- *reset:* señal de inicialización asíncrona para todos los biestables del circuito.
- $h2_{in}$*:* corriente de red una vez compensados los efectos de potencia

Salidas:

- $h2_{out}$: salida del regulador PI propuesto.

Este regulador se aplica sobre las señales obtenidas del sub-bloque previo: I_{d_sal} e I_{q_sal} por lo que será necesario mapear dos veces este regulador en el fichero final Top_Inversor.vhd, tal y como se verá en el próximo capítulo.

Como se ha visto en el cuarto capítulo de este Proyecto, la función de transferencia para el regulador PI es la siguiente:

$$PI(s) = Kp_{PI}\frac{1 + Ki_{PI}\cdot s}{Ki_{PI}\cdot s} \qquad (5.26)$$

Que tal como se mostró, en nuestro caso se particulariza a la siguiente ecuación:

$$PI(s) = 0.0141733\cdot\frac{1 + 0.00339366\cdot s}{0.00339366\cdot s} \qquad (5.27)$$

No obstante, esta expresión es para el dominio continuo, por lo que será necesario obtener los nuevos coeficientes para el caso discreto.
Así pues, la función de transferencia de primer orden en el dominio Z queda de la siguiente forma:

$$PI(z) = \frac{b_0\cdot z^N + b_1\cdot z^{N-1}}{a_0\cdot z^N + a_1\cdot z^{N-1}} \qquad (5.28)$$

Los coeficientes obtenidos para el control digital son:

a_0=1
a_1=-1
b0=0.01419
b1=-0.01416

Al ser $a_0 = 1$, la función de transferencia de este regulador puede ser expresada mediante la siguiente expresión en diferencias:

$$a_0\, h2_{out}[n] + a_1\, h2_{out}[n\text{-}1] = b_0\, h2_{in}[n] + b_1\, h2_{in}[n\text{-}1] \qquad (5.29)$$

Como condición inicial, se supondrá $h2_{in}[n]=0$.

5.5.2 Ruta de datos

En la siguiente figura se muestra el diseño digital del bloque que se describe en VHDL.

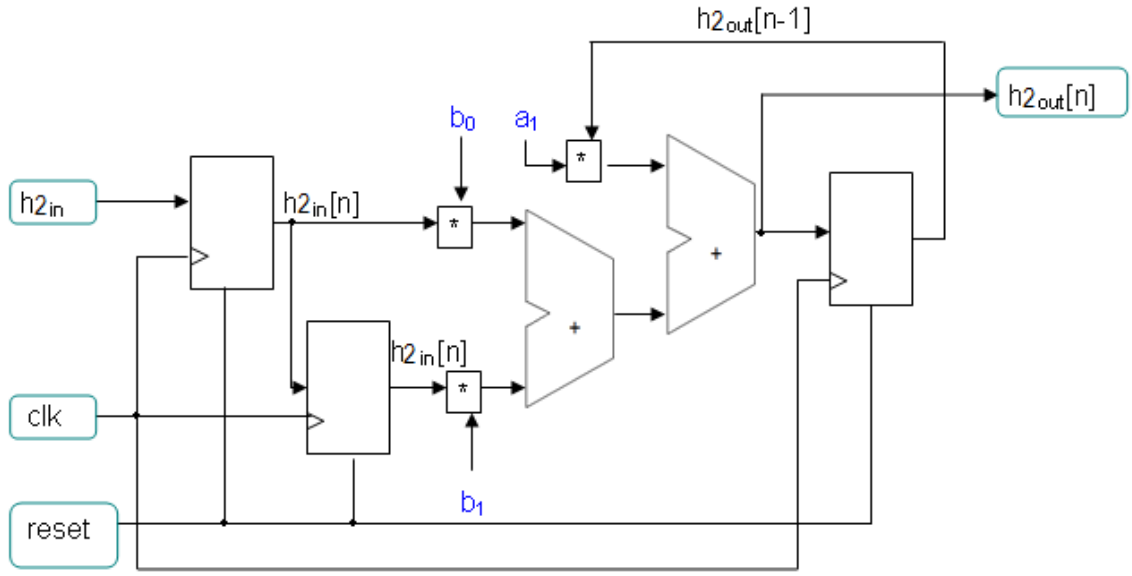


Figura 5.18. Ruta de datos del regulador.

5.5.3 Descripción en VHDL

Bibliotecas:
Una vez más se incluyen las mismas bibliotecas que en los sub-bloques previos.
Entidad:
Se declara la entidad **H2_filter**, donde se incluyen los genéricos reales correspondientes a las constantes descritas previamente. Las entradas de esta entidad son *clk* y *reset*, de tipo "std_logic", y *h2_in*, de tipo real. La salida, también de tipo real es *h2_out*.
Arquitectura:
La arquitectura H2 de la entidad ***H2_filter*** consta de cuatro registros que almacenarán los valores de las entradas y las salidas en el instante actual y en el instante previo.

5.5.4 Verificación de resultados

Una vez más se implementa un banco de pruebas donde se mapea la entidad ***H2_filter*** y se declara un proceso para la generación de la señal de reset, otro para la señal de reloj y un último proceso que permite una lectura en cada ciclo de reloj (de periodo 5µs) de las señales de entrada a leer del fichero obtenido mediante la simulación en PSIM.

A continuación se muestra la comparativa de resultados obtenidos mediante PSIM y Modelsim a la salida de este regulador, usando como señal de entrada I_{q_fil}, ya que si el resultado obtenido es correcto, lo será también aplicando el filtro sobre I_{d_fil}.

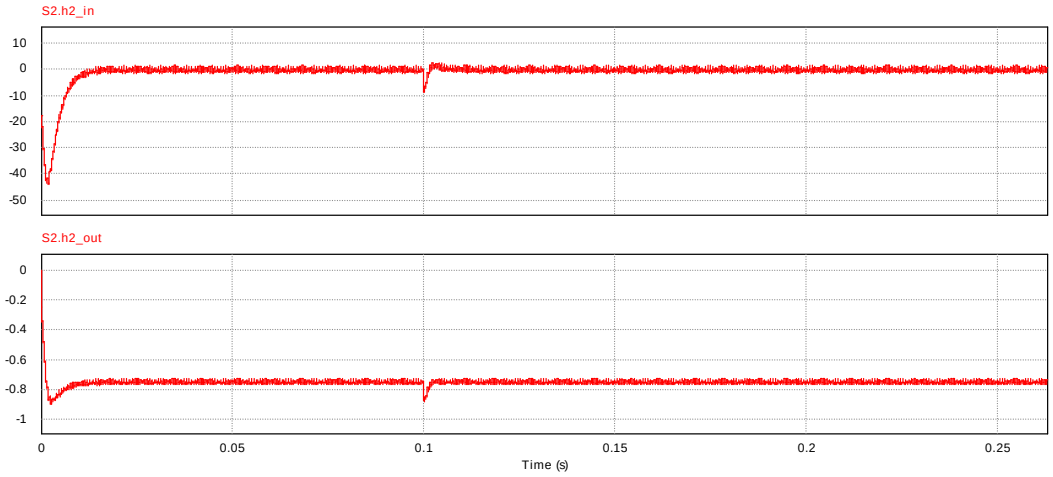


Figura 5.19. Entrada y salida del regulador simuladas mediante PSIM.

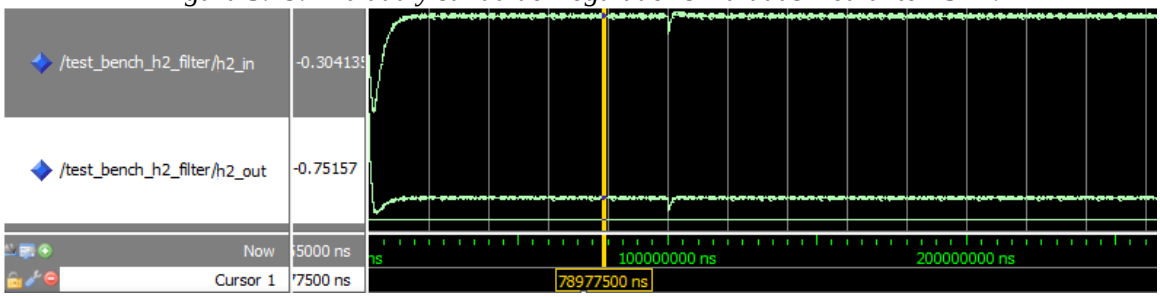


Figura 5.20.Entrada y salida del regulador obtenidas mediante Modelsim.

Comprobando los resultados obtenidos mediante Modelsim con el fichero de salida del regulador para PSIM, se comprueba que los resultados son esperados, estimando para ello el error relativo promediado para las primeras cincuenta muestras.

De esta forma, se obtiene un valor de $\bar{\varepsilon}$=0.89% a la salida del regulador PI, que nos permite pasar al siguiente bloque con la certeza de que todo lo implementado funciona correctamente.

5.6 Ruta_Datos2

5.6.1 Descripción

En este tercer sub-bloque denominado Ruta_Datos2, se implementa el control de las potencias activa y reactiva.

Entradas:

- *clk*: reloj de sincronismo del circuito para todos los biestables del mismo.
- *reset*: señal de inicialización asíncrona para todos los biestables del circuito.
- I_{d_sal}, I_{q_sal}, I_{d_fil}, I_{q_fil}: las entradas de este bloque son las señales obtenidas a la salida del bloque ***Ruta_Datos1***.

Salidas:

- d_{dqo_abc} y q_{dqo_abc}: señales desacopladas a inyectar a la red en el dominio *dq* y que serán las entradas al conversor dqo_abc.

Como se verá más adelante, a la hora de la integración del sistema final con PSIM, en este bloque entrará en juego una entrada adicional llamada *decoup* que se generará en el bloque final integrador llamado ***Top_inversor***. No obstante, para este sub-bloque será considerado como una constante, tal y como se explica en el apartado 5.6.2.

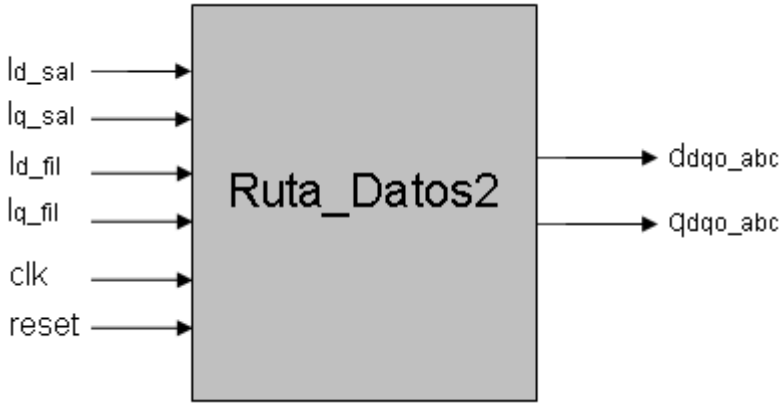


Figura 5.21. Entidad d Ruta_Datos2.

5.6.2 Ruta de datos

Como puede verse en la ruta de datos mostrada en la Figura 5.22, en este sub-bloque interviene el componente ***H2_filter*** correspondiente al regulador ya imlementado.
Además interviene el factor de desacoplo debido a la bobina. El cálculo de este valor (considerado como una constante para este sub-bloque) se calcula de la siguiente forma:

$$decoup = \frac{L \cdot w}{\left(Vdc_{sw} - Vdc_{offset}\right) / \mathrm{Re}lVdc} \qquad (5.30)$$

$$V_{dc\_offset} = Vin_{\max} \cdot \frac{1 - \left(Vo_{\max} - Vo_{\min}\right)}{Vin_{\max} - Vin_{\min}} \qquad (5.31)$$

$$\mathrm{Re}lVdc = \frac{Vo_{\max} - Vo_{\min}}{Vin_{\max} - Vin_{\min}} \qquad (5.32)$$

Con $Vo_{\max}$ y $Vo_{\min}$, tensión máxima y mínima respectivamente a la salida del conversor analógico/digital, y con $Vin_{\max}$ y $Vin_{\min}$ las tensión máxima y mínima respectivamente a la entrada del conversor analógico/digital.

Se obtiene el siguiente valor: *decoup* = 0.0014954119480036277.

A continuación se muestra la ruta de datos para el diseño VHDL de este sub-bloque.

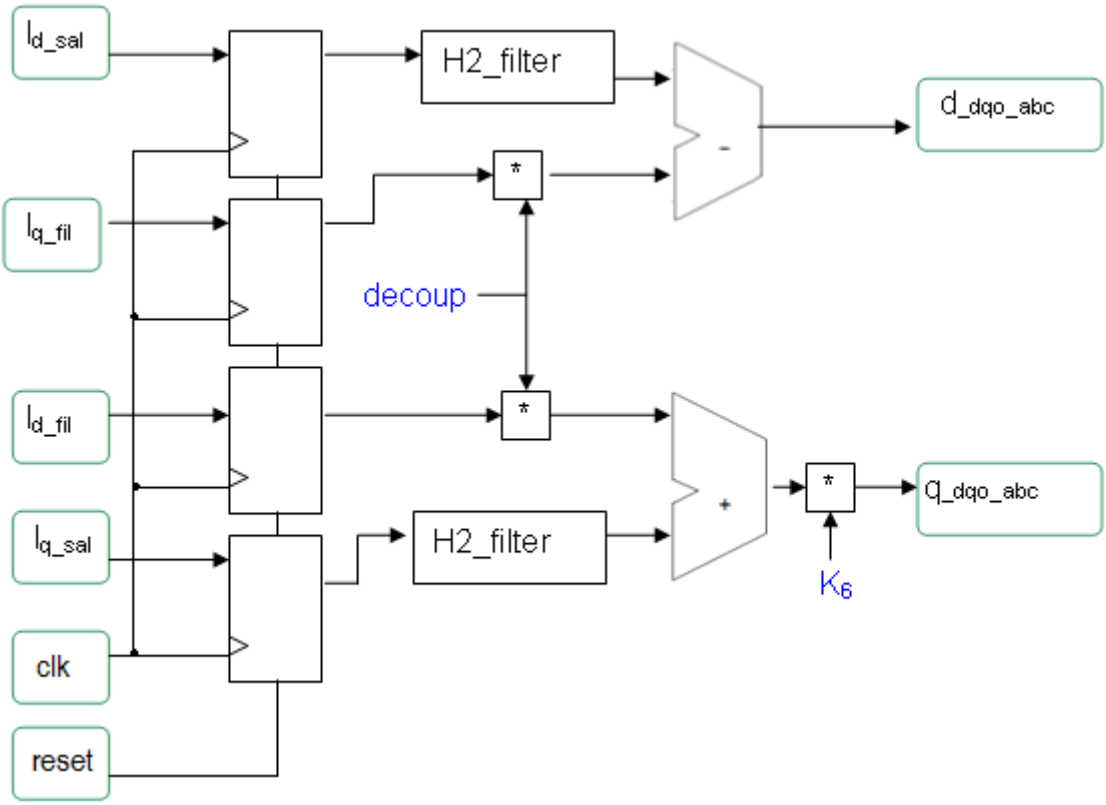


Figura 5.22. Ruta de datos de Ruta_Datos2.

5.6.3 Descripción en VHDL

Bibliotecas:
Se incluyen las mismas bibliotecas que en los sub-bloques previos.
Entidad:
Se declara la entidad ***Ruta_Datos2***. Las entradas de esta entidad son *clk* y *reset*, de tipo "*std_logic*", e I_{d_sal}, I_{q_sal}, I_{d_fil}, I_{q_fil}, *decoup*, de tipo real. Las salidas, también de tipo real, son $d_{\_dqo\_abc}$ y $q_{\_dqo\_abc}$.
Arquitectura:
La arquitectura Ruta2 de la entidad ***Ruta_Datos2*** consta de la declaración del componente ***H2_filter*** que será mapeado posteriormente y de seis registros que almacenarán los valores de las entradas y las salidas del bloque ***Ruta_Datos2***.

5.6.4 Verificación de resultados

Nuevamente, una vez implementado este bloque, se crea un banco de pruebas que permita comprobar el correcto uncionamiento del código desarrollado. En dicho banco de pruebas se mapea la entidad ***Ruta_Datos2*** y se declara un proceso para la generación de la señal de reset, otro para la señal de reloj y un último proceso que permite una lectura en cada ciclo de reloj de las señales de entrada a leer de los cuatro ficheros de entrada obtenidos mediante la simulación en PSIM.

Las salidas de este sub-bloque (d_{dqo_abc} y q_{dqo_abc}) obtenidas mediante PSIM y posteriormente mediante Modelsim, se muestran a continuación:

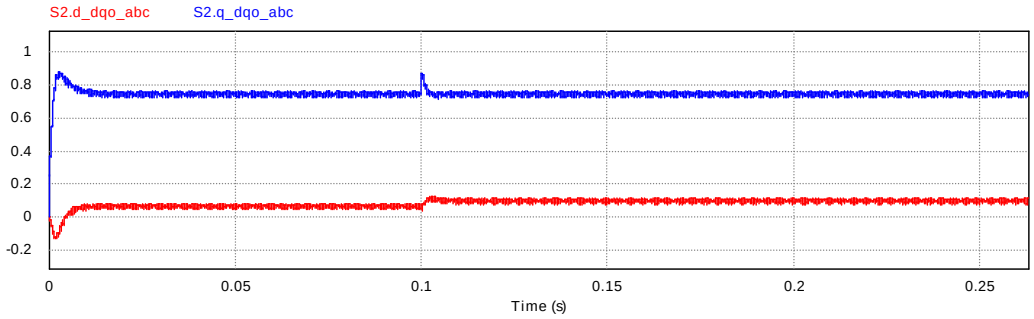


Figura 5.23. Salidas de Ruta_Datos2 simuladas mediante PSIM.

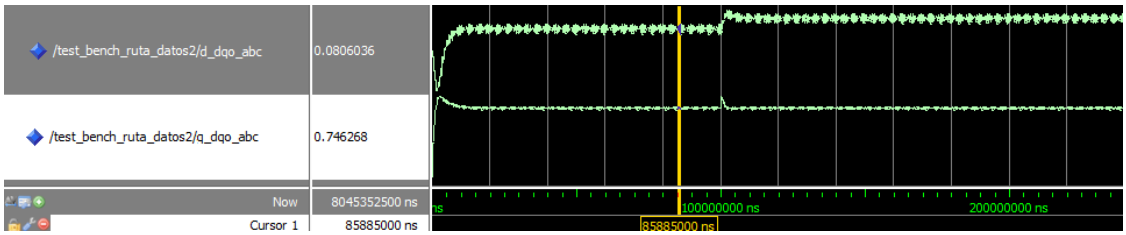


Figura 5.24. . Salidas de Ruta_Datos2 obtenidas mediante Modelsim.

Una vez más, se comprueba que efectivamente los valores obtenidos son correctos a partir de los resultados obtenidos mediante Modelsim y del fichero de salida para PSIM, calculando el error relativo y promediándolo para las primeras cincuenta muestras, con lo que se obtiene un valor de $\bar{\varepsilon}$=3.7% para d_{dqo_abc} y $\bar{\varepsilon}$=4.6% para q_{dqo_abc}. Así pues, se obtienen valores razonables que permiten intuir un correcto funcionamiento del actual sub-bloque.

5.7 dqo_abc

5.7.1 Descripción

Por último, se procede a la implementación del último sub-bloque que contendrá el boque final ***Top_Inversor***. Se trata de la transformación inversa de *Park* de referencia arbitraria que responde a la Ecuación 5.33:

$$T_{dqo\,/\,abc} = \begin{bmatrix} \cos(w\cdot t) & sen(w\cdot t) & 1 \\ \cos\left(w\cdot t - \frac{2\pi}{3}\right) & sen\left(w\cdot t - \frac{2\pi}{3}\right) & 1 \\ \cos\left(w\cdot t + \frac{2\pi}{3}\right) & sen\left(w\cdot t + \frac{2\pi}{3}\right) & 1 \end{bmatrix} \qquad (5.33)$$

Como ya se ha comentado, puesto que en este proyecto se está trabajando con números reales, no es necesario acudir a ninguna aproximación algorítmica que proporcione un resultado aproximado de las funciones seno y coseno.

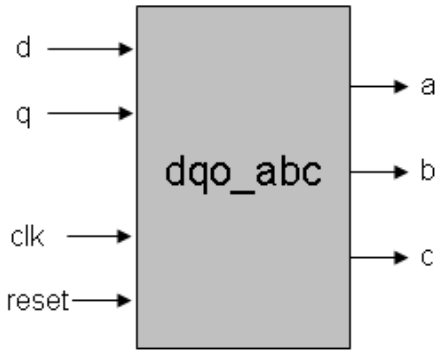


Figura 5.25. Entidad de la transformada de Park inversa.

Entradas:

- clk: reloj de sincronismo del circuito para todos los biestables del mismo.
- reset: señal de inicialización asíncrona para todos los biestables del circuito.
- d, q: tensión a inyectar a la red una vez se realice el cambio de referencia.

Salidas:

- a, b, c: tensión a inyectar a la red, que en la Figura 5.2 se denominan V_{mod_a}, V_{mod_b} *y* V_{mod_c}.

5.7.2 Ruta de datos

Para el caso del ***dqo_abc*** (tal y como se hizo para el ***abc_dqo***), se deja indicada la ruta de datos correspondiente a este bloque.

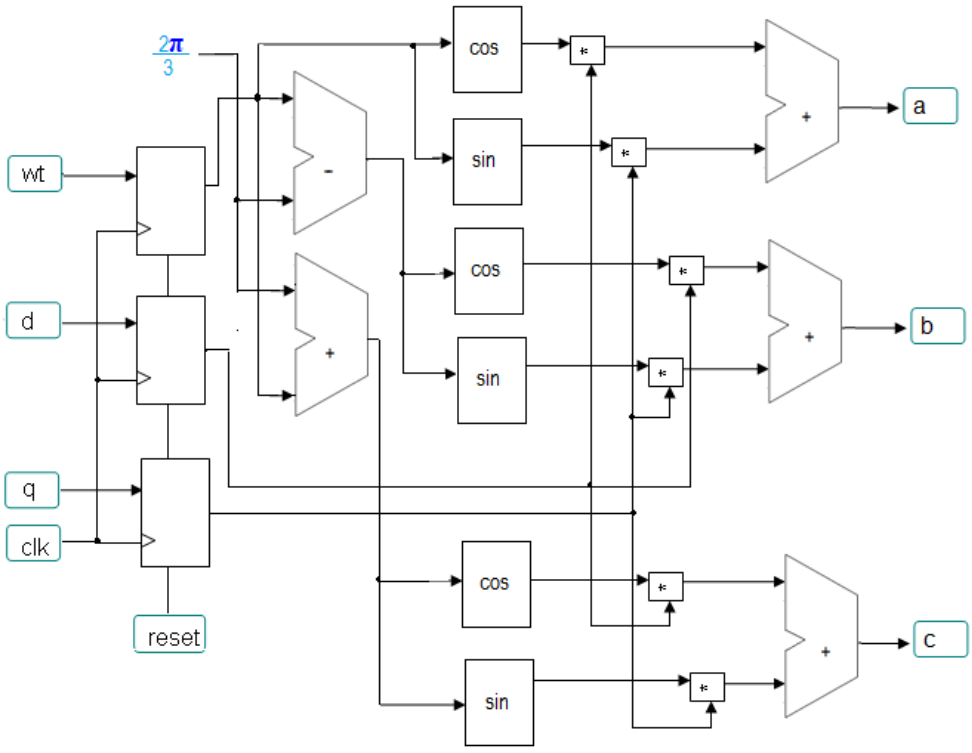


Figura 5.26. Ruta de datos simplificada para la transformada inversa de Park.

5.7.3 Descripción en VHDL

Bibliotecas:
Las ya especificadas anteriormente.
Entidad:
Se declara la entidad ***dqo_abc***, donde se declara un genérico real que será el coeficiente 2/3 que multiplica a π. Se declaran también los puertos de entrada y salida.
Las entradas clk y reset son de tipo *"std_logic"*, mientras que las entradas *d*, *q* y las salidas *a*, *b*, *c* serán de tipo real.
Arquitectura:
La arquitectura *d* de la entidad ***dqo_abc***, consta de tres registros que almacenarán los valores de las entradas (*d*, *q* y *wt*) necesarias para obtener cada una de las salidas.
Dentro de esta arquitectura se encuentra el proceso "biestable" cuya lista de sensibilidad incluye las señales de reset y clk que como ya se ha comentado, marcarán al proceso cuándo ejecutarse.
De esta forma, en cada ciclo de reloj se irán almacenando los valores de las tres señales pertenecientes a esta arquitectura.
Fuera ya del proceso definido, se realiza la asignación concurrente de las ecuaciones correspondientes al cálculo de *a*, *b* y *c* a las salidas de la entidad.

5.7.4 Verificación de resultados

Nuevamente con el fin de poder llevar a cabo la simulación en Modelsim, se ha implementado un banco de pruebas donde se mapea la entidad ***dqo_abc*** y se declaran tres procesos: uno para la generación de la señal de reset, otro para la señal de reloj y un último proceso que permite una lectura en cada ciclo de reloj de las señales de entrada *d*, *q*, *o* que se encuentran en los respectivos ficheros de entrada generados mediante PSIM. Dichos ficheros se encuentran localizados en el directorio donde se halla el proyecto implementado.

A continuación se muestran las salidas obtenidas para este sub-bloque.

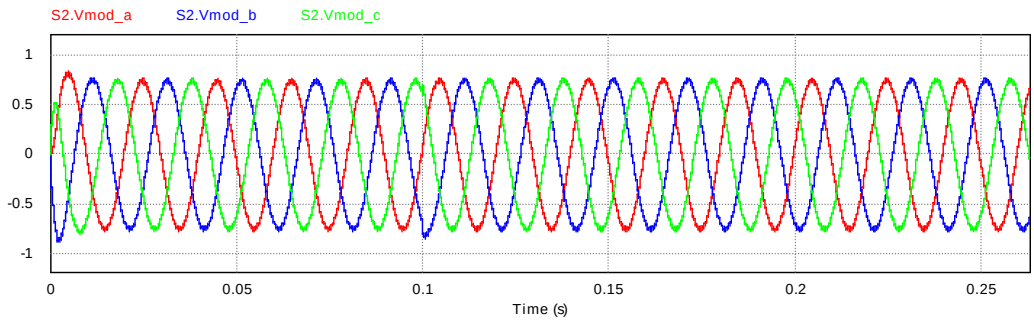


Figura 5.27. Salidas de dqo_abc simuladas mediante PSIM.

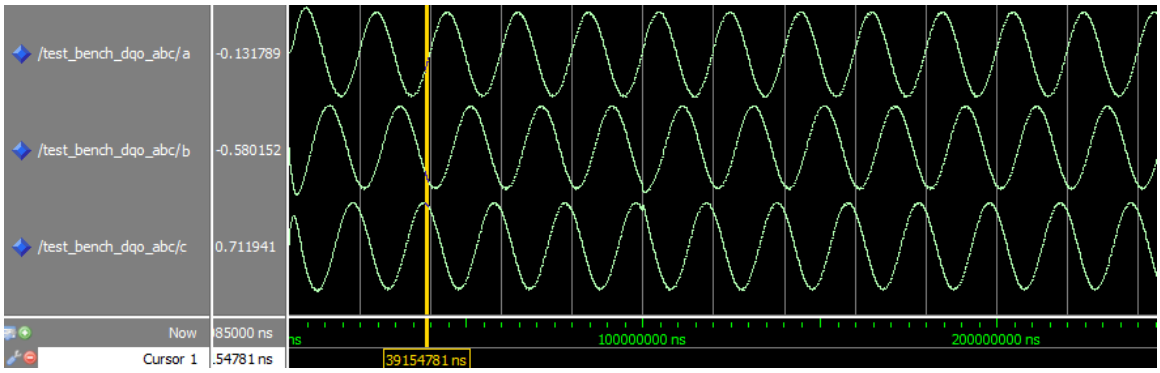


Figura 5.28. . Salidas de dqo_abc obtenidas mediante Modelsim.

Estas señales serán las que se utilicen en el modulador PWM, el cual será implementado en el bloque final ***Top_Inversor*** descrito en el próximo capítulo.

Por último, se comprueba una vez más que efectivamente los valores obtenidos son los deseados, calculando para ello, el error relativo en cada instante y promediándolo para las primeras cincuenta muestras, con lo que se obtiene un valor de $\bar{\varepsilon}$=1.87% para a, $\bar{\varepsilon}$=1.96% para b y $\bar{\varepsilon}$=1.92% para c. De estos valores, se deduce que la simulación del código VHDL desarrollado para este bloque también es correcta, por lo que se procede a la integración de todos los bloques, tal y como se describe en el próximo capítulo.

Capítulo 6

Resultados de simulación

6.1 Introducción

A lo largo de este capítulo se va a llevar a cabo la sistematización de la puesta en marcha de la simulación conjunta analógico-digital propuesta en este proyecto.
Para ello, se propone la siguiente organización del proyecto, tal y como ha sido desarrollada la aplicación.

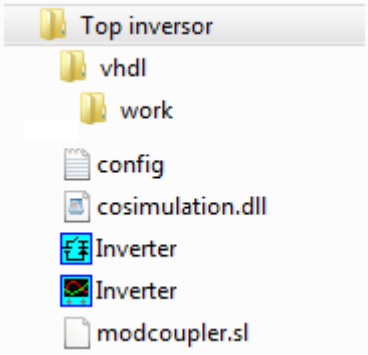


Figura 6.1.Organización del proyecto.

- **Top inversor:** Directorio del diseño descrito en VHDL que contiene el código y el archivo de inicio de Modelsim, que no es más que el original que presenta ModelSim en su directorio de instalación, añadiéndole la ruta de trabajo:
    - ;MI DIRECTORIO DE TRABAJO
    - Work = ./vhdl/work

  El proyecto Modelsim es el que incluye el código VHDL del diseño. Será un proyecto activo desde el punto de vista de la depuración del diseño digital. El propósito de este proyecto es compilar el diseño completo a simular junto a PSIM y servir de plataforma para el diseño del control digital

  Para la versión software usada (ModelSim SE 6.3j sobre Windows 7), las operaciones básicas para compilar el diseño son las siguientes:
    - ✓ Se crea un proyecto de Modelsim "*File->New->Project"* con la localización adecuada (vhdl, en la organización propuesta) y fijando el parametro *'default library name'* a "*work"*.
    - ✓ A continuaciones *añaden todas las unidades de diseño VHDL necesarias*
    - ✓ Por último, se establece el orden de compilación y se lanza la misma mediante *"Compile All"*.

- **Cosimulation.dll:** que usa PSIM y cuyo nombre debe ir en concordancia con el correspondiente *dll*_block del diseño en PSIM y debe ser copiada en el directorio raiz del diseño, para poder ser utilizado como *dll_block.*

- **Modcoupler.sl:** Archivo generado tras la compilación y enlazado del código.

- **Inverter.sch:** Diseño en esquemático del circuito PSIM, en el que se incluye:
    - Circuito de potencia.
    - El ***dll_block***.
    - Fichero de configuración, donde se fijan los parámetros

Una vez incluidos estos bloques en PSIM, se obtiene le esquemático necesario para realizar la simulación conjunta.

6.2 Top_Inversor

6.2.1 Circuito inversor en PSIM

A continuación se plantea el circuito de potencia con su control en PSIM, pero esta vez incluyendo el bloque *dll* para realizar la simulación conjunta.
Se va a llevar a cabo también la implementación VHDL del circuito de control digital final, a partir de los sub-bloques ya implemetados y explicados en el capítulo previo. Al bloque final que integra todos estos sub-bloques y que además añade la modulación PWM, se le ha llamado ***Top_Inversor***.

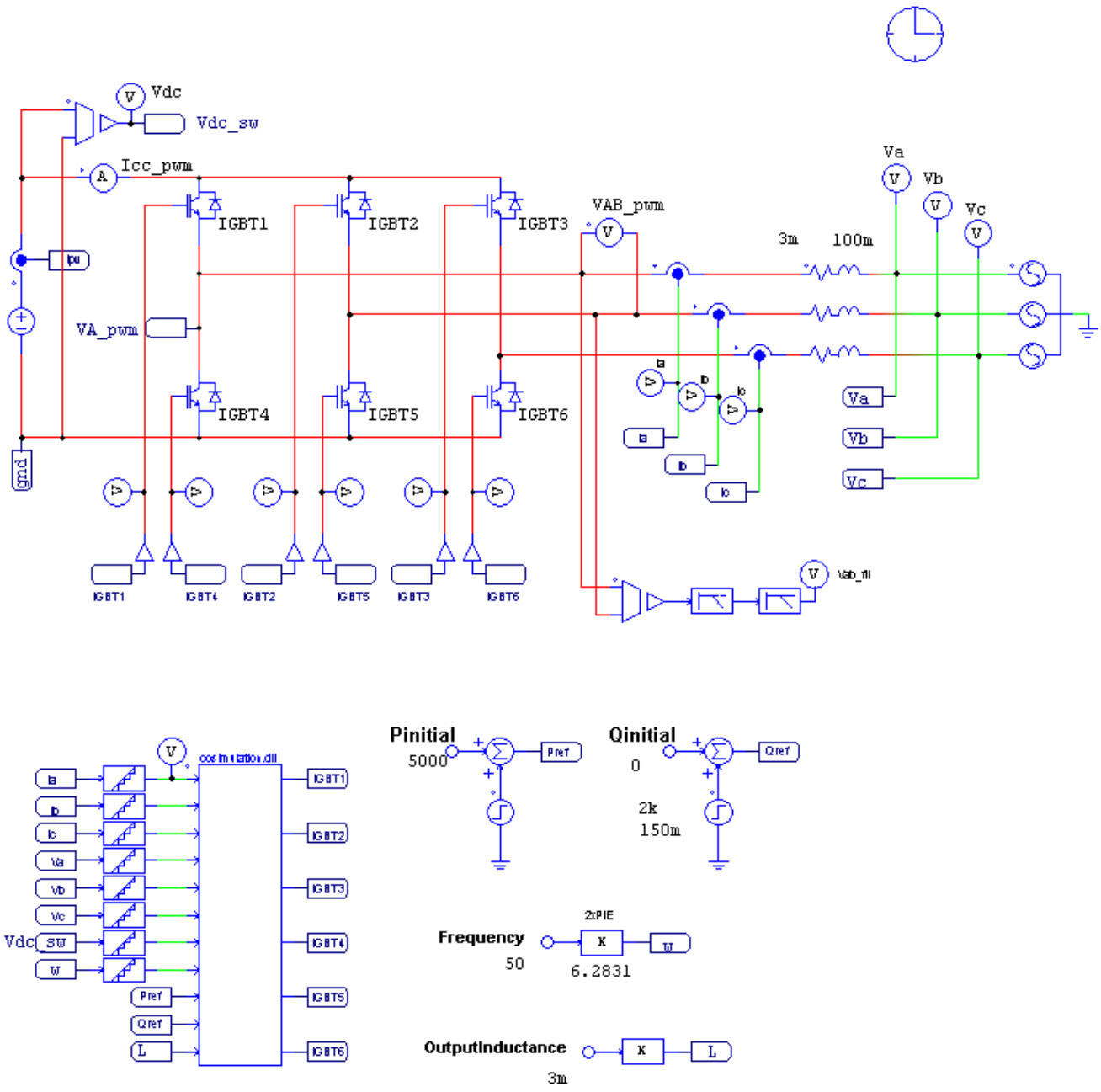


Figura 6.2 .Inversor de potencia trifásico conectado a red.

Como puede apreciarse en la Figura 6.2, el bloque *dll* llamado *cosimulation* consta de once entradas y seis salidas:

Entradas:

- Corrientes sinusoidales i_a, i_b *e* i_c desfasadas 120º entre ellas.
- Tensiones sinusoidales v_a, v_b y v_c también desfasadas 120º entre ellas.
- Tensión de contínua v_{cd_sw}.
- Frecuencia de entrada al ***dll*** en radianes, que será: $w=2\cdot\pi\cdot f =2\cdot\pi\cdot(50Hz)=314.16$ rad.
- Inductancia L de la bobina, que se usará (junto con v_{cd_sw} y w) para el cálculo del factor de desacoplo (*decoup*), tal como se ha visto en el capítulo anterior.
- Potencias de referencia activa y reactiva: P_{ref} y Q_{ref} que se emplearán para llevar a cabo el control de potencia.

Salidas:

- Señales de control *IGBT1, IGBT2, IGBT3, IGBT4, IGBT5 e IGBT6* que harán que se activen y desactiven cada uno de los transistores IGBT.

6.2.2 Descripción

La implementación VHDL de este bloque final, se llevará a cabo en el bloque Top_Inversor, que contendrá mapeados de cada uno de los componentes ya definidos en los sub-bloques explicados en el capítulo previo, así como la implementación del modulador PWM.

La entidad para este ***Top_Inversor***, se muestra en la siguiente figura.

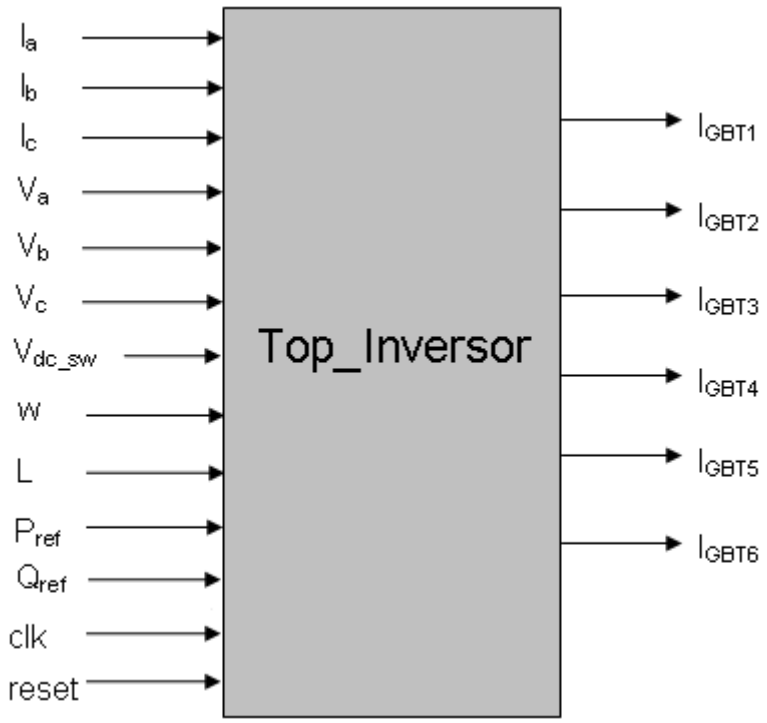


Figura 6.3.Entidad de Top_Inversor.

Como se ve, las entradas y salidas de la entidad VHDL coinciden con la del bloque ***dll***. Además de estas entradas y salidas, en este caso hay que añadir también las señales *clk* y *reset*, al igual que en el resto de los sub-bloques que componen este inversor.

6.2.3 Ruta de datos

En la siguiente figura se muestra la ruta de datos para el diseño digital de ***Top_Inversor*** en VHDL.

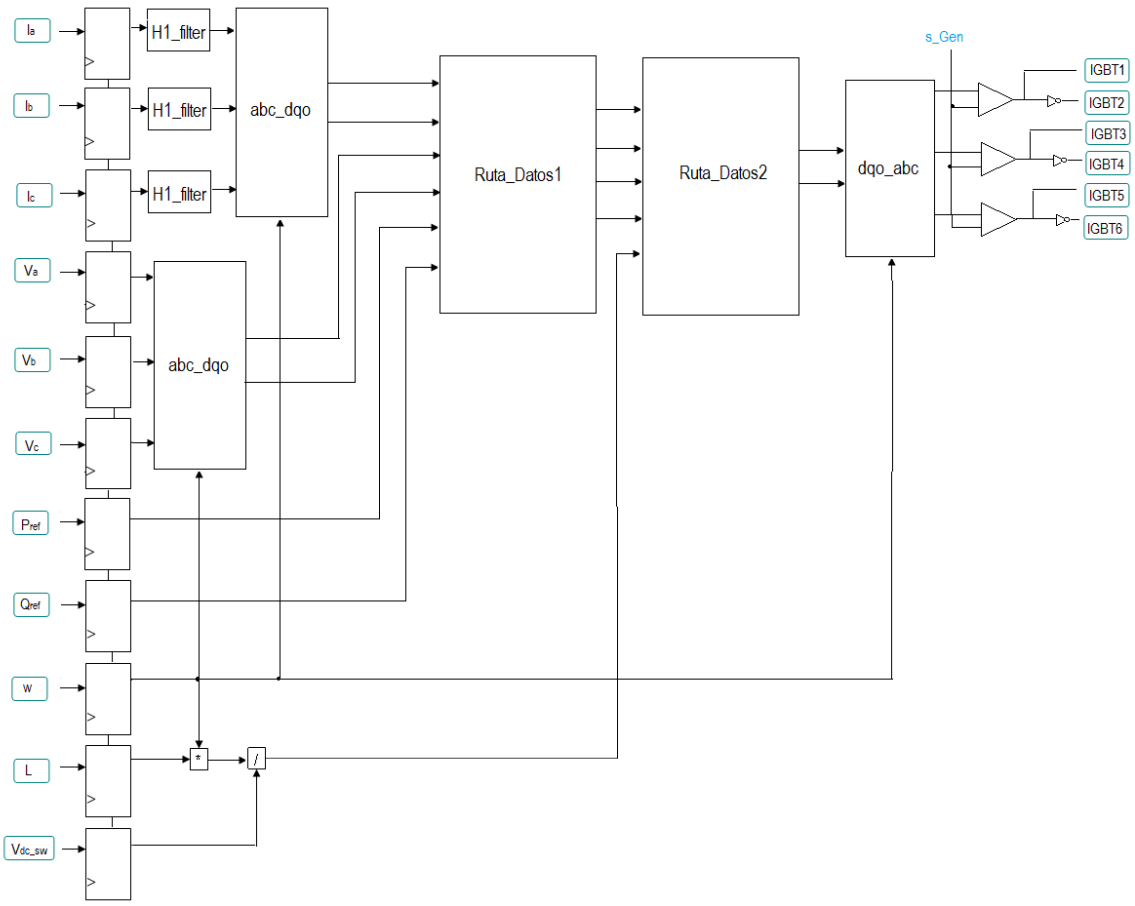


Figura 6.4. Ruta de datos de Top_Inversor.

6.2.4 Descripción en VHDL

Bibliotecas:
Una vez más se incluyen las mismas bibliotecas que en los sub-bloques previos.
Entidad:
Dentro de este campo se declaran las entradas y salidas ya indicadas y donde tanto las dos entradas *clk*, *reset* como las seis salidas serán de tipo *"std_logic"*, mientras que el resto de entradas serán de tipo *"std_logic_vector"* para que sean compatibles con las señales que se reciban de PSIM y que serán convertidas a reales dentro de este bloque.

```
s_Ia      <= real(conv_integer(Ia));
s_Ib      <= real(conv_integer(Ib));
s_Ic      <= real(conv_integer(Ic));
s_Va      <= real(conv_integer(Va));
s_Vb      <= real(conv_integer(Vb));
s_Vc      <= real(conv_integer(Vc));
s_w       <= real(conv_integer(w))*10000.0/2.0**31;
s_Pref    <= real(conv_integer(P_ref));
s_Qref    <= real(conv_integer(Q_ref));
```

Arquitectura:

En la arquitectura Top de la entidad Top_Inversor, se declaran todos los componentes necesarios tal y como se muestra en la Figura 6.4 para posteriormente mapearlos con las señales de entrada/salida del fichero ***Top_Inversor***.

```
CMP: process(s_Vmod_a, s_Vmod_b, s_Vmod_c, s_Gen,clk)
 begin
  if s_Vmod_a > s_Gen then
     IGBT1 <= '1';
     IGBT4 <= '0';
  else
     IGBT1 <= '0';
     IGBT4 <= '1';
  end if;

  if s_Vmod_b > s_Gen then
     IGBT2 <= '1';
     IGBT5 <= '0';
  else
     IGBT2 <= '0';
     IGBT5 <= '1';
  end if;
  if s_Vmod_c > s_Gen then
     IGBT3 <= '1';
     IGBT6 <= '0';
  else
     IGBT3 <= '0';
     IGBT6 <= '1';
  end if;
end PROCESS;
```

La señal *s_Gen* se genera también en otro proceso llamado GEN, tal como se muestra a continuación:

```
GEN:process
  constant period: time := 660 us;
  constant amplitude: real:= 2.0;
  constant resolution: integer := 1000;
begin
  s_Gen <= -1.0;
  for i in 1 to resolution loop
    s_Gen <= s_Gen+amplitude/real(resolution);
    wait for period/resolution/2;
  end loop;
  for i in 1 to resolution loop
    s_Gen <= s_Gen-amplitude/real(resolution);
    wait for period/resolution/2;
  end loop;
end process;
```

6.2.5 Verificación de resultados

Por último, se llevará a cabo la verificación del funcionamiento final del inversor de potencia. En primer lugar, en la Figura 6.5, pueden observarse los saltos de potencia activa y reactiva y cómo se regulan.
La primera gráfica muestra la potencia activa de referencia (P_{ref}) y la potencia activa instantánea (p_{dq}), que como puede verse es pulsante debido al desequilibrio.
La segunda gráfica muestra la potencia reactiva de referencia (Q_{ref}) y la potencia activa instantánea (q_{dq}) que, de igual manera, es pulsante.

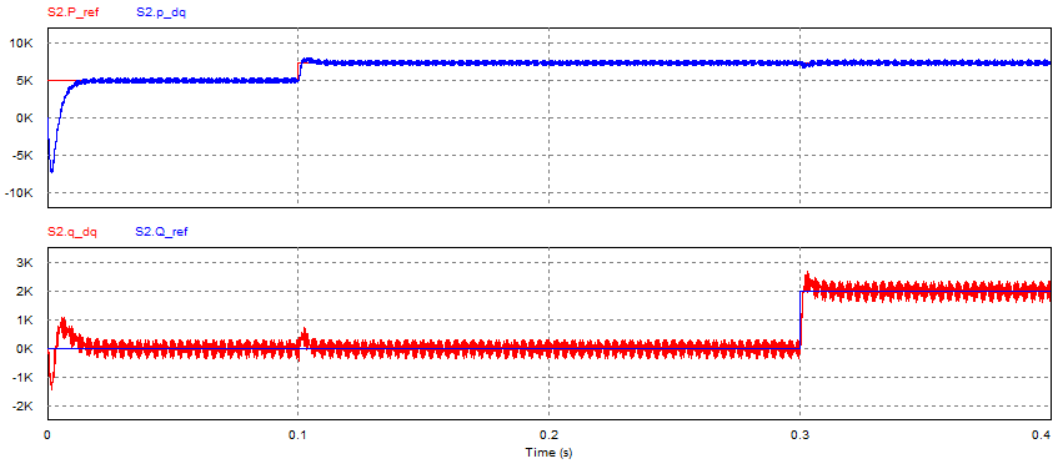


Figura 6.5. Regulación de potencias activa y reactiva.

En la Figura 6.6 se aprecia cómo ante un aumento en el escalón P_{ref} se produce un aumento de corriente. Además se puede apreciar que el transitorio de corriente tiene una duración inferior a un ciclo de red.
Por otro lado, se aprecia un desfase en la corriente ante el salto en Q_{ref}.

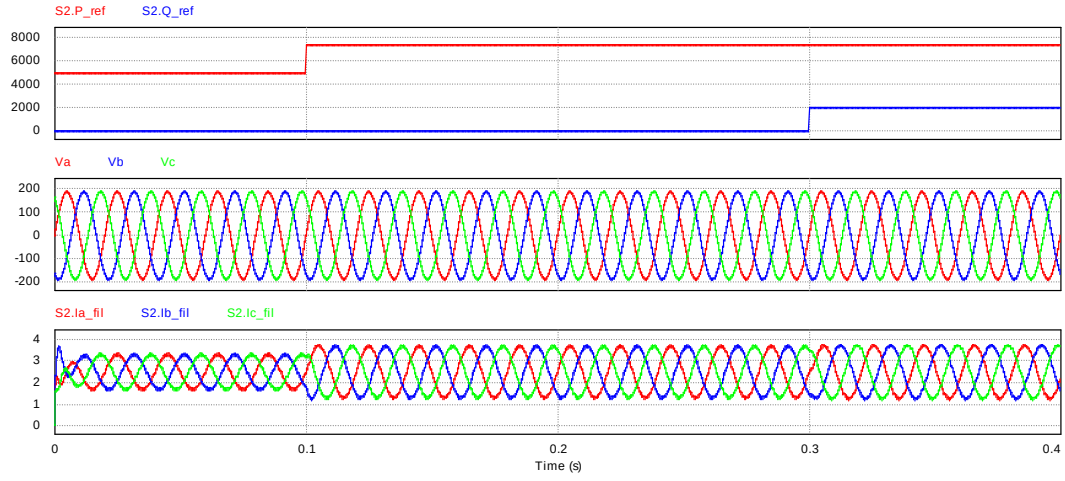


Figura 6.6.Corriente y tensión ante escalones de potencia..

Con el fin de mostrar el correcto funcionamiento del sistema de control, el error relativo promediado para cinco mil muestras, obtenidos mediante las Ecuaciónes 5.3 y 5.4 indicadas previamente y partiendo de los valores resultantes de la simulación analógica

con PSIM y la cosimulación analógico-digital para la señal V_{AB}, indicada en las Figuras 4.2 y 6.2, es del 1.36%, valor muy aceptable.

Capítulo 7

Conclusiones y trabajo futuro

7.1 Conclusiones

En este trabajo se ha estudiado el modelo de un inversor y se ha diseñado una estructura de control, mediante la transformada de *Park* que translada el problema a un plano síncrono d-q y haciendo uso de reguladores PI para eliminar armónicos de la tensión de salida del inversor.

Posteriormente y partiendo del circuito de potencia y sus especificaciones, se ha llevado el sistema de control al plano digital, haciendo uso de conversores analógico-digitales y realizando las operaciones pertinentes sobre los coeficientes de las funciones de transferencia para obtener los nuevos coeficientes para las funciones de transferencia discretas.

Una vez definido el sistema, se ha procedido al diseño digital de cada uno de los bloques implicados en el control, a su descripción en VHDL y a la validación del correcto funcionamiento de cada uno de estos bloques haciendo uso de bancos de pruebas.

Finalmente, se ha validado el diseño digital del control mediante su integración en un entorno de cosimulación analógioa-digital, con las herramientas comericales Modelsim y PSIM. Para ello se ha descrito un fichero VHDL que agrupa cada uno de los bloques desarrollados y que compone el funcionamiento del sistema de control digital propuesto previamente mediante PSIM. Posteriormente, se ha procedido a la simulación conjunta

del sistema de control con el circuito de potencia, contrastando los resutados de simulación mediante PSIM con los obtenidos de la simulación conjunta PSIM-Modelsim.

Con este Proyecto Final de Carrera, se ha probado una metodología de diseño mixta, para los circuitos de conversión de energía eléctrica controlados mediante dispositivos electrónicos. El control digital es una solución barata, eficiente y de bajo consumo que puede proporcionar ventajas competitivas a los dispositivos electrónicos de potencia que lo incluyan.

La necesidad de una metodología de diseño de este tipo de sistemas es una realidad y con este trabajo se ha aplicado un método basado en la cosimulación con herramientas comerciales Los resultados obtenidos permiten ser optimistas ante las posibilidades que se abren en este campo.

7.2 Líneas de trabajo futuro

Partiendo del trabajo realizado a lo largo de este Proyecto Final de Carrera, surgen varias líneas de trabajo que pueden ser tratadas en trabajos futuros:

- **Descripción de modelos sintetizables manteniendo las posibilidades de cosimulación.** Como ya se ha comentado previamente, es facil obtener operaciones equivalentes sintetizables para el caso del producto. En adición, el presente documento se ha mostrado una pequeña orientación de cómo podría llevarse a cabo la implementación sintetizable para el caso de la transformada de *Park* (directa e inversa), haciendo uso de desarrollos de *Taylor* que permitan una aproximación de las funciones seno y coseno necesarias en estos bloques.
  No obstante, sería interesante la implementación de un divisor sintetizable para usar en Ruta_Datos1 y que permita una implementación sintetizable del sistema completo.
- **Uso de lazo de seguimiento de fase** en marco de referencia síncrono en el sistema de control, que permita inyectar corriente en fase con la red.

Presupuesto

En esta sección se presenta el presupuesto del proyecto. En él se contempla la duración de las distintas fases y tareas, y se incluye un desglose de costes de personal, costes de material y costes totales.

1. Tareas

Fase 1: Planificación.

- Estudio de los sistemas inversores de potencia.
  - Duración: 25 días.
- Estudio del algoritmo de control elegido.
  - Duración: 25 días.
- Estudio de las tecnologías necesarias.
  - Duración: 25 días.

Fase 2: Desarrollo.

- Análisis y diseño inicial.
  - Duración: 20 días.
- Implementación del sistema.
  - Duración: 60 días.
- Pruebas unitarias.
  - Duración: 60 días.
- Fase de integración.
  - Duración: 30 días.

- Evaluación de la aplicación.
  - Duración: 15 días.

Fase 3: Documentación.

- Memoria del Proyecto Final de Carrera.
  - Duración: 50 días.
- Preparación de la presentación.
  - Duración: 5 días.

2. Recursos

Para la realización del presente proyecto se han utilizado los siguientes recursos:

- **Recursos software:**
  - Licencia Microsoft Office 2007**: 200 €**
  - Editor de programación Notepad ++: **0 €**
  - Licencia Matlab: **6000 €**
  - Mantenimiento anual PSIM 9.03 Trial: **500 €**
  - Mantenimiento anual Modelsim SE Plus 6.3j: **500 €**
- **Recursos hardware:**
  - Ordenador portátil: **800 €**
- **Recursos humanos:**
  - Tutores de proyecto.
  - Desarrollador.

  Las horas dedicadas al proyecto han sido una media de cinco horas diarias, contemplando semanas de siete días.
  - Coste de un Ingeniero: **30 €/hora.**

3. Resumen de costes

El coste de recursos humanos de la aplicación se resume en la Tabla 8.1.

TAREA	*DÍAS*	*IMPORTE*
Fase 1	75	11.250 €
Fase 2	165	24.750 €
Fase 3	55	8.250 €
Subtotal	295	44.250 €

Tabla 8.1.Detalle de costes de Recursos Humanos del Proyecto.

El coste total del sistema se presenta en la tabla 8.2:

CONCEPTO	*IMPORTE*
Recursos Humanos	44.250
Recursos Software	7.200
Recursos Hardware	800
Subtotal	52.250
(18% IVA)	9.405
TOTAL	61.655

Tabla 8.2. Detalle de Coste Total del Proyecto.

El presupuesto total de este proyecto asciende a la cantidad de SESENTA Y UN MIL SEISCIENTOS CINCUENTA Y CINCO EUROS.

Madrid a 18 de Octubre de 2011

Fdo. Beatriz Brogeras García.

Bibliografía

[1] http://www.monografias.com/trabajos27/analogico-y-digital/analogico-y-digital.shtml, Accedido en Julio 2011.

[2] Rashid, M.H. 'Electrónica de potencia: Circuitos, dispositivos y aplicaciones'.Ed: Pearson – Prentice Hall. 2004. Capítulos 3 y 10.

[3] http://www.netrino.com/Embedded-Systems/How-To/PWM-Pulse-Width-Modulation, Accedido en Julio 2011.

[4] http://www.ni.com/fpga_technology/esa/, Accedido en Julio 2011.

[5] Rodrigálvarez, D. 'Metodología de diseño de convertidores de potencia con control digital basada en simulación analógico-digital conjunta'. Proyecto Fin de Carrera, Universidad Carlos III de Madrid, 2007.

[6] Zainalabedin Navabi. 'VHDL: Analysis and Modeling of Digital Systems'. Mc Graw Hill. 1992.

[7] Estrella, D. 'Diseño de un Inversor Monofásico Autónomo de Baja Frecuencia Ajustable mediante Bus DC'. Proyecto fin de Carrera, Universidad Carlos II de Madrid, 2009.

[8] Rodríguez, C.A. 'Diseño e implementación de prácticas para el laboratorio de electrónica industrial: inversor monofásico'. Trabajo de grado, Pontificia Universidad Javeriana, 2004.

[9] Lucena, C., 'Implementación digital de elementos de control y sincronización de conexión a red para convertidores electrónicos de potencia'. Trabajo Fin de Máster, Universidad Carlos II de Madrid, 2011.

[10] http://tec.upc.es/el/TEMA-1%20EP%20(v1).pdf, Accedido en Julio 2011.

[11] http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/salvatori_a_m/capitulo4.pdf, Accedido en Julio 2011.

[12] Hassaine, L. 'Implementación de un control digital de potencia activa y reactiva para inversores. Aplicación a sistemas fotovoltaicos conectados a red'. Tesis Doctoral, Universidad Carlos III de Madrid, 2010.

[13] http://es.wikipedia.org/wiki/Hist%C3%A9resis, Accedido Septiembre 2011.

[14] Mohan, N., Underland, T.M., Robins, W. P. 'Power Electronics: Converters, applications and design'. Third Edition. Ed. John Wiley & Sons, Inc. 2003.

[15] http://es.wikipedia.org/wiki/Teorema_de_Fortescue, Accedido Agosto 2011.

[16] Gonzalez Longatt, F.M., 'Entendiendo la Transformación de *Park*'. Junio 2004.

[17] http://upcommons.upc.edu/pfc/bitstream/2099.1/2696/5/36106-5.pdf, Accedido Septiembre 2011.

[18] Ferrero, F.J. 'Descripción de circuitos digitales mediante VHDL'. Universidad de Oviedo. Mayo 1999.

[19] Modelsim SE User's Manual Version 6.0, July 20, 2004.

[20] PSIM User's Guide, Version 6.0 Rev.1 January 22, 2004.

[21] PSIM simulation software Tutorial: How to use DLL Block, 2004.

[22] Modelsim SE Foreign Language Interface (FLI) Reference Version 6.0, July 20, 2004.

[23] Barrado, A. and Lázaro, A. 'Problemas de Electrónica de Potencia'. (Pearson-Prentice Hall, 2007).

[24] Bueno Peña, E. 'Optimización del comportamiento de un Convertidor de tres niveles NPC Conectado a la Red Eléctrica'. Tesis Doctoral', Universidad de Alcalá, 2005.

[25] De Castro, A. 'Aplicación del control digital basado en hardware específico para convertidores de potencia conmutados'. Tesis Doctoral, Universidad Politécnica de Madrid, 2003.

[26] 'IEEE Standard VHDL Language Reference Manual'. Published by the Institute of Electrical and Electronics Engineers, 1994.

[27] Roncero, P. L., García, A., Feliu, V., and García, P. 'control en lazo cerrado para eliminación de armónicos en inversores fuente de tensión'. Ciudad Real, 8 Sept. 2004.

[28] Alepuz, S. 'Aportación al Control del Convertidr CC/CA de Tres Niveles'. Tesis Doctoral, Universitat Politecnica de Catalunya, 2004.

[29] http://www.uma.es/investigadores/grupos/electronica_potencia, Accedido en Julio 2011.

[30] http://catarina.udlap.mx/u_dl_a/tales/documentos/meie/herber_r_jj/capitulo1.pdf, Accedido en Julio 2011.

Anexos

abc_dqo.vhd sintetizable

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;
--In case of vectors with pure binary representation
use ieee.std_logic_signed.all;
--BASIC package
use work.basic.all;

entity abc_dqo is
generic ( a1: std_logic_vector((abcdqoCoefWidth-1) downto 0) :="0000001010101010";-- 2/3
          a2: std_logic_vector((abcdqoCoefWidth-1) downto 0) := "0000000101010101";--1/3
          a3: std_logic_vector((abcdqoCoefWidth-1) downto 0) := "0000000010101010";-- 1/3!
          a4: std_logic_vector((abcdqoCoefWidth-1) downto 0) := "0000001000000000";-- 1/2!
         ..a5: std_logic_vector((abcdqoCoefWidth-1) downto 0) := "0000010000000000";--1
          pi: std_logic_vector((abcdqoCoefWidth-1) downto 0) := "0000110010010000");-- pi

port(clk, reset : in std_logic;
     a,b,c: in std_logic_vector (abcdqoInputWidth-1 downto 0);
     wt: in std_logic_vector (abcdqoCoefWidth-1 downto 0);
     d,q,o: out std_logic_vector (abcdqoOutputWidth-1 downto 0
end abc_dqo;

architecture a of abc_dqo is
  signal sa, sb, sc: std_logic_vector (abcdqoInputWidth-1 downto 0);
  signal sangle1, sangle2, sangle3: std_logic_vector (abcdqoInputWidth downto 0);
  signal scos1, scos2, scos3, ssin1, ssin2, ssin3 : std_logic_vector (abcdqoTrigWidth-1 downto 0);
--Establecer numer de bits

begin
P_BIEST: process(reset, clk)
   begin
      if reset='1' then
         sa <= (others => '0');
         sb <= (others => '0');
         sc <= (others => '0');
         sangle1 <= (others => '0');
         sangle2 <= (others => '0');
```

```
        sangle3 <= (others => '0');
        scos1 <= (others => '0');
        scos2 <= (others => '0');
        scos3 <= (others => '0');
        ssin1 <= (others => '0');
        ssin2 <= (others => '0');
ssin3 <= (others => '0');

elsif clk'event and clk='1' then
        sa <= a;
        sb <= b;
        sc <= c;
        sangle1 <= wt;
        sangle2 <= '0'& wt - '0'&(a1*pi); -- wt-2pi/3
        angle3 <= '0'& wt +'0'&(a1*pi); -- wt+2pi/3
        scos1 <= '0' & a4 -'0' & (a3*sangle1*sangle1);
        scos2 <= '0' & a4 -'0' & (a3*sangle2*sangle2);
        scos3 <= '0' & a4 -'0' & (a3*sangle3*sangle3);
        ssin1 <= '0' & sangle1 - '0' & (a3*sangle1*sangle1*sangle1
        ssin2 <= '0' & sangle2 - '0' & (a3*sangle2*sangle2*sangle2);
        ssin3 <= '0' & sangle3 - '0' & (a3*sangle3*sangle3*sangle3);
end if;
end process;

d <= a1*(('0'&(scos1*sa))+('0'&( ('0'&(scos2*sb))+('0'&(scos3*sc)))));
q <= a1*(('0'&(ssin1*sa))+('0'&( ('0'&(ssin2*sb))+('0'&(ssin3*sc)))));
o <= a2*(('0'& sa)+('0'&(('0'& sb)+ ('0' & sc))));
```

H1_filter.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;

entity H1_filter is

--Real coefficients
generic ( a1: real := -0.9752;
        b0: real := 0.01242;
        b1: real := 0.01242);

port(clk, reset : in std_logic;
    --In
    H1_in : in real;
    --Out
    H1_out : out real);
end H1_filter;

architecture H1 of H1_filter is
```

```
   signal act_input, prev_input, act_output, prev_output: real :=0.0;

begin
P_BIEST_1: process(reset, clk)
   begin
      if reset='1' then
         act_input <= 0.0;
         prev_input <= 0.0;
      elsif clk'event and clk='1' then
         act_input <= H1_in;
         prev_input <= act_input;
      end if;
end process;

P_BIEST_2: process (reset,clk)
begin
   if reset = '1' then
      prev_output <= 0.0;
   elsif clk'event and clk='1' then
      prev_output <= act_output;
   end if;
end process;
act_output <= (b0*act_input + b1*prev_input)-(a1*prev_output);
H1_out <= act_output;
end H1;
```

Test_bench_H1_filter.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;

---------------------------------
--Files reading
use std.textio.all;
use ieee.std_logic_textio.all;
---------------------------------

ENTITY test_bench_H1_filter IS
END test_bench_H1_filter;
ARCHITECTURE test OF test_bench_H1_filter IS

--File statement. TEXT type
FILE Ia: TEXT IS IN "Ia.txt";
```

```
component h1_filter
      port(clk, reset:in std_logic;
           H1_in: in real;
           H1_out: out real);
end component;

signal s_clk, s_reset: std_logic;
signal s_H1_in: real;
signal s_H1_out: real;
constant period: time :=5 us;

BEGIN
uut: h1_filter port map (clk => s_clk,
                         reset => s_reset,
                         H1_in => s_H1_in,
                         H1_out => s_H1_out);

clk_gen: PROCESS
BEGIN
   s_clk <='0';
   wait for period/2;
   s_clk <= '1';
   wait for period/2;
END PROCESS clk_gen;

reset_gen: PROCESS
BEGIN
    s_reset <='1';
    wait for period;
    s_reset <='0';
    wait for 4000000*period;
 --   assert false
  --     report "End of the simulation"
  --     severity failure;
END PROCESS reset_gen;

in_gen: PROCESS
--Stimulus
VARIABLE i : real;
VARIABLE linea : LINE;
BEGIN

IF not (ENDFILE(Ia)) THEN
    --To read a line of the file
    READLINE (Ia, linea);
    READ (linea, i);
END IF;
s_H1_in <= i;
wait for period;
```

```
END PROCESS in_gen;

END test;
```

abc_dqo.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;
--In order to use common real constants, common real functions, and real trascendental functions.
use ieee.math_real.all;

entity abc_dqo is
  generic(
    a1    : real := 0.66666666666666666666666666667; -- 2/3
    a2    : real := 0.33333333333333333333333333333); -- 1/3
  port(
    clk    : in  std_logic;
    reset  : in  std_logic;
    a      : in  real;
    b      : in  real;
    c      : in  real;
    w      : in  real;
    d      : out real;
    q      : out real);
end abc_dqo;

architecture a of abc_dqo is
  signal sa    : real := 0.1;
  signal sb    : real := 0.1;
  signal sc    : real := 0.1;
  signal sw    : real := 0.1;
begin
  sa   <= a;
  sb   <= b;
  sc   <= c;
  sw   <= w;
  d                                                    <=          a1*((cos(sw)*sa)+(cos(sw-(a1*MATH_PI))*sb)+(cos(sw+(a1*MATH_PI))*sc));
  q    <= a1*((sin(sw)*sa)+(sin(sw-(a1*MATH_PI))*sb)+(sin(sw+(a1*MATH_PI))*sc));
end a;
```

Test_bench_abc_dqo.vhd

```
library IEEE;
```

```
--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;

---------------------------------
--Files reading
use std.textio.all;
use ieee.std_logic_textio.all;
---------------------------------

ENTITY test_bench_abc_dqo IS
END test_bench_abc_dqo;
ARCHITECTURE test OF test_bench_abc_dqo IS

--File statement. TEXT type
FILE a: TEXT IS IN "Ia_fil.txt";
FILE b: TEXT IS IN "Ib_fil.txt";
FILE c: TEXT IS IN "Ic_fil.txt";
FILE wt: TEXT IS IN "wt.txt";

component abc_dqo
     port(clk, reset:in std_logic;
          a,b,c,wt: in real;
          d,q: out real);
end component;

signal s_clk, s_reset: std_logic;
signal s_a,s_b,s_c, s_wt: real;
signal s_d,s_q: real;
constant period: time :=5 us;

BEGIN
uut: abc_dqo port map (clk   => s_clk,
                       reset => s_reset,
                       a     => s_a,
                       b     => s_b,
                       c     => s_c,
                       wt    => s_wt,
                       d     => s_d,
                       q     => s_q);

clk_gen: PROCESS
BEGIN
  s_clk <='0';
  wait for period/2;
  s_clk <= '1';
  wait for period/2;
END PROCESS clk_gen;
```

```
reset_gen: PROCESS
BEGIN
   s_reset <='1';
   wait for period;
   s_reset <='0';
   wait for 4000000*period;
 --   assert false
 --     report "End of the simulation"
 --     severity failure;
END PROCESS reset_gen;

in_gen: PROCESS
--Stimulus
VARIABLE ia : real;
VARIABLE linea_a : LINE;
VARIABLE ib : real;
VARIABLE linea_b : LINE;
VARIABLE ic : real;
VARIABLE linea_c : LINE;
VARIABLE iwt : real;
VARIABLE linea_wt : LINE;
BEGIN

IF not (ENDFILE(a)) THEN
   --To read a line of the file
   READLINE (a, linea_a);
   READ (linea_a, ia);
END IF;
s_a <= ia;

IF not (ENDFILE(b)) THEN
   --To read a line of the file
   READLINE (b, linea_b);
   READ (linea_b, ib);
END IF;
s_b <= ib;

IF not (ENDFILE(c)) THEN
   --To read a line of the file
   READLINE (c, linea_c);
   READ (linea_c, ic);
END IF;
s_c <= ic;

IF not (ENDFILE(wt)) THEN
   --To read a line of the file
   READLINE (wt, linea_wt);
   READ (linea_wt, iwt);
END IF;
```

```
s_wt <= iwt;

wait for period;

END PROCESS in_gen;

END test;
```

Ruta_Datos1.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;

entity Ruta_Datos1 is
  generic (
    k1       : real := 0.02734375;
    k2       : real := -0.02734375;
    k3       : real := 0.1220703125;
    k4       : real := -0.1220703125;
    k5       : real := 0.66666666666666666666666666666667);
  port(
    clk      : in  std_logic;
    reset    : in  std_logic;
    Id       : in  real;
    Iq       : in  real;
    Vd       : in  real;
    Vq       : in  real;
    Pref     : in  real;
    Qref     : in  real;
    Id_sal   : out real;
    Iq_sal   : out real;
    Id_fil   : out real;
    Iq_fil   : out real);
end Ruta_Datos1;

architecture Ruta1 of Ruta_Datos1 is
  signal sid    : real := 1.0;
  signal siq    : real := 1.0;
  signal svd    : real := 1.0;
  signal svq    : real := 1.0;
  signal spref  : real := 0.0;
  signal sqref  : real := 0.0;
  signal a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q : real := 1.0;
 begin
  P_BIEST: process(reset, clk)
```

```
 begin
   if reset='1' then
     sid <= 0.0;
     siq <= 0.0;
     svd <= 1.0;
     svq <= 1.0;
     spref <= 5000.0;
     sqref <= 0.0;
   elsif clk'event and clk='1' then
     sid <= Id;
     siq <= Iq;
     svd <= Vd;
     svq <= Vq;
     spref <= Pref;
     sqref <= Qref;
   end if;
 end process;
 p      <= k1*sid;
 q      <= k2*siq;
 Id_fil <= p;
 Iq_fil <= q;
 a      <= k3*svd;
 b      <= k4*svq;
 c      <= a*spref;
 d      <= b*sqref;
 e      <= b*spref;
 f      <= a*sqref;
 g      <= c+d;
 h      <= e-f;
 i      <= a*a;
 j      <= b*b;
 k      <= i+j;
 l      <= g/k;
 m      <= h/k;
 n      <= l*k5;
 o      <= m*k5;
 Id_sal <= n-p;
 Iq_sal <= o-q;
end Ruta1;
```

Test_bench_Ruta_Datos1.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;
```

```
---------------------------------
--Files reading
use std.textio.all;
use ieee.std_logic_textio.all;
---------------------------------


ENTITY test_bench_Ruta_Datos1 IS
END test_bench_Ruta_Datos1;
ARCHITECTURE testRD1 OF test_bench_Ruta_Datos1 IS

--File statement. TEXT type
FILE p_ref: TEXT IS IN "Pref.txt";
FILE q_ref: TEXT IS IN "Qref.txt";
FILE vd: TEXT IS IN "Vd.txt";
FILE vq: TEXT IS IN "Vq.txt";
FILE id: TEXT IS IN "Id.txt";
FILE iq: TEXT IS IN "Iq.txt";

component Ruta_Datos1
     port(clk, reset:in std_logic;
          Id, Iq, Vd, Vq, Pref, Qref: in real; --Pref, Qref,
          Id_sal, Iq_sal : out real;
          Id_fil, Iq_fil : out real);

end component;

signal s_clk, s_reset: std_logic;
signal s_Id, s_Iq, s_Vd, s_Vq, s_Pref, s_Qref: real; --s_Pref, s_Qref,
signal s_Id_sal, s_Iq_sal, s_Id_fil, s_Iq_fil : real; --Creo que puedo prescindir de esto
constant period: time :=5 us;

BEGIN
uuut: ruta_datos1 port map (clk => s_clk,
                        reset => s_reset,
                        Id=> s_Id,
                        Iq=>s_Iq,
                        Vd=>s_Vd,
                        Vq=>s_Vq,
                        Pref=>s_Pref,
                        Qref=>s_Qref,
                        Id_sal => s_Id_sal,
                        Iq_sal => s_Iq_sal,
                        Id_fil => s_Id_fil,
                        Iq_fil => s_Iq_fil); --Pref => s_Pref, Qref => s_Qref,

clk_gen: PROCESS
BEGIN
  s_clk <='0';
  wait for period/2;
```

```
  s_clk <= '1';
  wait for period/2;
END PROCESS clk_gen;

reset_gen: PROCESS
BEGIN
   s_reset <='1';
   wait for period;
   s_reset <='0';
   wait for 4000000*period;
 --  assert false
 --     report "End of the simulation"
 --     severity failure;
END PROCESS reset_gen;

in_gen: PROCESS
--Stimulus
VARIABLE ipref : real;
VARIABLE linea_pref : LINE;
VARIABLE iqref : real;
VARIABLE linea_qref : LINE;
VARIABLE ivd : real;
VARIABLE linea_vd : LINE;
VARIABLE ivq : real;
VARIABLE linea_vq : LINE;
VARIABLE iid : real;
VARIABLE linea_id : LINE;
VARIABLE iiq : real;
VARIABLE linea_iq : LINE;
BEGIN

IF not (ENDFILE(p_ref)) THEN
   --To read a line of the file
   READLINE (p_ref, linea_pref);
   --Read data interpreted as std_logic_vector type
   READ (linea_pref, ipref);
END IF;
s_Pref <= ipref;
IF not (ENDFILE(q_ref)) THEN
   --To read a line of the file
   READLINE (q_ref, linea_qref);
   --Read data interpreted as std_logic_vector type
   READ (linea_qref, iqref);
END IF;
s_Qref  <= iqref;
IF not (ENDFILE(vd)) THEN
   --To read a line of the file
   READLINE (vd, linea_vd);
   --Read data interpreted as std_logic_vector type
   READ (linea_vd, ivd);
```

```
END IF;
s_Vd <= ivd;

IF not (ENDFILE(vq)) THEN
   --To read a line of the file
   READLINE (vq, linea_vq);
   --Read data interpreted as std_logic_vector type
   READ (linea_vq, ivq);
END IF;
s_Vq <= ivq;

IF not (ENDFILE(id)) THEN
   --To read a line of the file
   READLINE (id, linea_id);
   --Read data interpreted as std_logic_vector type
   READ (linea_id, iid);
END IF;
s_Id <= iid;

IF not (ENDFILE(iq)) THEN
   --To read a line of the file
   READLINE (iq, linea_iq);
   --Read data interpreted as std_logic_vector type
   READ (linea_iq, iiq);
END IF;
s_Iq <= iiq;

wait for period;

END PROCESS in_gen;

END testRD1;
```

H2_filter.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;
--BASIC package
--use work.basic.all;

entity H2_filter is
  --Real coefficients
  generic(
    a1: real := -1.0;
    b0: real :=  0.01419;
```

```
   b1: real := -0.01416);
  port(
   clk      : in  std_logic;
   reset    : in  std_logic;
   --In
   H2_in    : in  real;
   --Out
   H2_out   : out real);
end H2_filter;

architecture H2 of H2_filter is
  signal act_input, prev_input, act_output, prev_output: real :=0.0;
begin
  P_BIEST_1: process(reset, clk)
  begin
   if reset='1' then
     act_input <= 0.0;
     prev_input <= 0.0;
   elsif clk'event and clk='1' then
     if H2_in > 50.0 then
       act_input <= 50.0;
     elsif H2_in < -50.0 then
       act_input <= -50.0;
     else
       act_input <= H2_in;
     end if;
     prev_input <= act_input;
   end if;
  end process;
  P_BIEST_2: process (reset,clk)
  begin
   if reset = '1' then
     prev_output <= 0.0;
   elsif clk'event and clk='1' then
     prev_output <= act_output;
   end if;
  end process;
  act_output <= (b0*act_input + b1*prev_input)-(a1*prev_output);
  H2_out     <= act_output;
end H2;
```

Test_bench_H2_filter.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;
```

```
---------------------------------
--Lectura de ficheros
use std.textio.all;
use ieee.std_logic_textio.all;
---------------------------------

ENTITY test_bench_H2_filter IS
END test_bench_H2_filter;
ARCHITECTURE test OF test_bench_H2_filter IS

--File statement. TEXT type
FILE Iq_sal_fil: TEXT IS IN "Iq_sal_fil.txt";

component h2_filter
     port(clk, reset:in std_logic;
          H2_in: in real;
          H2_out: out real);
end component;

signal s_clk, s_reset: std_logic;
signal s_H2_in: real;
signal s_H2_out: real;
constant period: time :=5 us;

BEGIN
uut: h2_filter port map (clk => s_clk, reset => s_reset, H2_in => s_H2_in, H2_out =>
s_H2_out);

clk_gen: PROCESS
BEGIN
   s_clk <='0';
   wait for period/2;
   s_clk <= '1';
   wait for period/2;
END PROCESS clk_gen;

reset_gen: PROCESS
BEGIN
    s_reset <='1';
    wait for period;
    s_reset <='0';
    wait for 4000000*period;
 --   assert false
  --     report "End of the simulation"
  --     severity failure;
END PROCESS reset_gen;

in_gen: PROCESS
--Stimulus
```

```
VARIABLE i : real;
VARIABLE linea : LINE;
BEGIN

IF not (ENDFILE(Iq_sal_fil)) THEN
   --To read a line of the file
   READLINE (Iq_sal_fil, linea);
   READ (linea, i);
END IF;
s_H2_in <= i;
wait for period;

END PROCESS in_gen;

END test;
```

Ruta_Datos2.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;

entity Ruta_Datos2 is
  generic(
    k6       : real := -1.0);
  port(
    clk      : in  std_logic;
    reset    : in  std_logic;
    Id_sal   : in  real;
    Iq_sal   : in  real;
    Id_fil   : in  real;
    Iq_fil   : in  real;
    d_dqo_abc : out real;
    q_dqo_abc : out real;
    decoup   : in  real);  -- Ahora es entrada (se genera en Top_inversor)

end Ruta_Datos2;

architecture Ruta2 of Ruta_Datos2 is
  component h2_filter
    generic(
      a1: real := -1.0;
      b0: real :=  0.01419;
      b1: real := -0.01416);
    port(
      clk      : in  std_logic;
```

```
    reset    : in  std_logic;
    H2_in    : in  real;
    H2_out   : out real);
  end component;
  signal s_Id_sal   : real := 0.0;
  signal s_Iq_sal   : real := 0.0;
  signal s_Id_fil   : real := 0.0;
  signal s_Iq_fil   : real := 0.0;
  signal s_H2_out1  : real := 0.0;
  signal s_H2_out2  : real := 0.0;
begin
  uut1: h2_filter
  port map (
    clk     => clk,
    reset   => reset,
    H2_in   => s_Id_sal,
    H2_out  => s_H2_out1);

  uut2: h2_filter
  port map (
    clk     => clk,
    reset   => reset,
    H2_in   => s_Iq_sal,
    H2_out  => s_H2_out2);

  P_BIEST: process(reset, clk)
  begin
    if reset='1' then
      s_Id_sal <= 0.0;
      s_Iq_sal <= 0.0;
      s_Id_fil <= 0.0;
      s_Iq_fil <= 0.0;
    elsif clk'event and clk='1' then
      s_Id_sal <= Id_sal;
      s_Iq_sal <= Iq_sal;
      s_Id_fil <= Id_fil;
      s_Iq_fil <= Iq_fil;
    end if;
  end process;
  d_dqo_abc <=  s_H2_out1 - (decoup*s_Iq_fil);
  q_dqo_abc <= (s_H2_out2 + (decoup*s_Id_fil))*K6;
end Ruta2;
```

Test_bench_Ruta_Datos2

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
```

```
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;

---------------------------------
--Lectura de ficheros
use std.textio.all;
use ieee.std_logic_textio.all;
---------------------------------


ENTITY test_bench_Ruta_Datos2 IS
END test_bench_Ruta_Datos2;
ARCHITECTURE testRD2 OF test_bench_Ruta_Datos2 IS

--File statement. TEXT type
FILE Id_sal: TEXT IS IN "Id_sal.txt";
FILE Iq_sal: TEXT IS IN "Iq_sal.txt";
FILE Id_fil: TEXT IS IN "Id_fil.txt";
FILE Iq_fil: TEXT IS IN "Iq_fil.txt";

component Ruta_Datos2
     port(clk, reset:in std_logic;
          Id_sal, Iq_sal: in real;
          Id_fil, Iq_fil : in real;
          d_dqo_abc : out real;
          q_dqo_abc : out real);

end component;

signal s_clk, s_reset: std_logic;
signal s_Id_sal, s_Iq_sal: real;
signal s_Id_fil, s_Iq_fil : real;
signal s_d_dqo_abc, s_q_dqo_abc : real;
constant period: time :=5 us;


BEGIN

uuut: ruta_datos2 port map (clk => s_clk,
                        reset => s_reset,
                        Id_sal => s_Id_sal,
                        Iq_sal => s_Iq_sal,
                        Id_fil=> s_Id_fil,
                        Iq_fil=>s_Iq_fil,
                        d_dqo_abc => s_d_dqo_abc,
                        q_dqo_abc => s_q_dqo_abc);

clk_gen: PROCESS
BEGIN
  s_clk <= '0';--
```

```
   wait for period/2;
   s_clk <= '1';--
   wait for period/2;
END PROCESS clk_gen;

reset_gen: PROCESS
BEGIN
    s_reset <='1';
    wait for period;
    s_reset <='0';
    wait for 4000000*period;
 --   assert false
  --     report "End of the simulation"
  --     severity failure;
END PROCESS reset_gen;

in_gen: PROCESS
--Stimulus
VARIABLE i_Id_sal : real;
VARIABLE linea_Id_sal : LINE;
VARIABLE i_Iq_sal : real;
VARIABLE linea_Iq_sal : LINE;
VARIABLE i_Id_fil : real;
VARIABLE linea_Id_fil : LINE;
VARIABLE i_Iq_fil : real;
VARIABLE linea_Iq_fil : LINE;
BEGIN

IF not (ENDFILE(Id_sal)) THEN
    --To read a line of the file
    READLINE (Id_sal, linea_Id_sal);
    --Read data interpreted as std_logic_vector type
    READ (linea_Id_sal, i_Id_sal);
END IF;
s_Id_sal <= i_Id_sal;

IF not (ENDFILE(Iq_sal)) THEN
    --To read a line of the file
    READLINE (Iq_sal, linea_Iq_sal);
    --Read data interpreted as std_logic_vector type
    READ (linea_Iq_sal, i_Iq_sal);
END IF;
s_Iq_sal <= i_Iq_sal;

IF not (ENDFILE(Id_fil)) THEN
    --To read a line of the file
    READLINE (Id_fil, linea_Id_fil);
    --Read data interpreted as std_logic_vector type
    READ (linea_Id_fil, i_Id_fil);
END IF;
```

```
s_Id_fil <= i_Id_fil;

IF not (ENDFILE(Iq_fil)) THEN
   --To read a line of the file
   READLINE (Iq_fil, linea_Iq_fil);
   --Read data interpreted as std_logic_vector type
   READ (linea_Iq_fil, i_Iq_fil);
END IF;
s_Iq_fil <= i_Iq_fil;

wait for period;

END PROCESS in_gen;

END testRD2;
```

dqo_abc.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;
--BASIC package
--use work.basic.all;
use ieee.math_real.all;

entity dqo_abc is
  generic(
    a1          : real := 0.666666666666666666666666667); -- 2/3
  port(
    clk         : in  std_logic;
    reset       : in  std_logic;
    d           : in  real;
    q           : in  real;
    w           : in  real;
    a           : out real;
    b           : out real;
    c           : out real);
end dqo_abc;

architecture d of dqo_abc is
  signal sd  : real := 0.0;
  signal sq  : real := 0.0;
  signal sw  : real := 0.0;
begin
  P_BIEST: process(reset, clk)
  begin
```

```
    if reset='1' then
      sd <= 0.0;
      sq <= 0.0;
      sw  <= 0.0;
    elsif clk'event and clk='1' then
      sd <= d;
      sq <= q;
      sw  <=w;
    end if;
  end process;

  a <=(cos(sw)*sd)+(sin(sw)*sq)+(0.0);--'o' vale '-1'
  b <=(cos(sw-(a1*MATH_PI))*sd)+(sin(sw-(a1*MATH_PI))*sq)+(0.0);
  c <=(cos(sw+(a1*MATH_PI))*sd)+(sin(sw+(a1*MATH_PI))*sq)+(0.0);
end d;
```

Test_bench_dqo_abc.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;

---------------------------------
--Lectura de ficheros
use std.textio.all;
use ieee.std_logic_textio.all;
---------------------------------

ENTITY test_bench_dqo_abc IS
END test_bench_dqo_abc;
ARCHITECTURE test OF test_bench_dqo_abc IS

--File statement. TEXT type
FILE d: TEXT IS IN "d.txt";
FILE q: TEXT IS IN "q.txt";
--FILE o: TEXT IS IN "o.txt";
FILE wt: TEXT IS IN "wt.txt";

component dqo_abc
     port(clk, reset:in std_logic;
          d,q,wt: in real;
          a,b,c: out real);
end component;

signal s_clk, s_reset: std_logic;
signal s_d,s_q,s_wt: real;
```

```
signal s_a,s_b,s_c: real;
constant period: time :=5 us;

BEGIN
uut: dqo_abc port map (clk => s_clk,
                       reset => s_reset,
                       d => s_d,
                       q=>s_q,
                       wt => s_wt,
                       a=>s_a,
                       b=> s_b,
                       c=>s_c);

clk_gen: PROCESS
BEGIN
   s_clk <='0';
   wait for period/2;
   s_clk <= '1';
   wait for period/2;
END PROCESS clk_gen;

reset_gen: PROCESS
BEGIN
    s_reset <='1';
    wait for period;
    s_reset <='0';
    wait for 4000000*period;
 --   assert false
  --     report "End of the simulation"
  --     severity failure;
END PROCESS reset_gen;

in_gen: PROCESS
--Stimulus
VARIABLE id : real;
VARIABLE linea_d : LINE;
VARIABLE iq : real;
VARIABLE linea_q : LINE;
VARIABLE iwt : real;
VARIABLE linea_wt : LINE;
BEGIN

IF not (ENDFILE(d)) THEN
    --To read a line of the file
    READLINE (d, linea_d);
    READ (linea_d, id);
END IF;
s_d <= id;

IF not (ENDFILE(q)) THEN
```

```
   --To read a line of the file
   READLINE (q, linea_q);
   READ (linea_q, iq);
END IF;
s_q <= iq;

IF not (ENDFILE(wt)) THEN
   --To read a line of the file
   READLINE (wt, linea_wt);
   READ (linea_wt, iwt);
END IF;
s_wt <= iwt;

wait for period;

END PROCESS in_gen;

END test;
```

Top_Inversor.vhd

```
library IEEE;

--In order to use type
use ieee.std_logic_1164.all;
--In order to use defined arithmetic functions (sum, diff, product)
use ieee.std_logic_arith.all;



use ieee.std_logic_unsigned.all;



-- Anado la señal Decoup que se calcula a partir de tres entradas
-- L
-- w
-- Vdc_sw
-- Siguiendo la ecuacion (L*W)/((Vdc_sw-Vdc_offset)/0.025)
-- Vdc_offset se toma como constante

entity Top_Inversor is

port(clk, reset    : in  std_logic;
     Ia, Ib, Ic    : in  std_logic_vector(31 downto 0);
     Va, Vb, Vc    : in  std_logic_vector(31 downto 0);
     Vdc_sw        : in  std_logic_vector(31 downto 0);
     w             : in  std_logic_vector(31 downto 0);
     P_ref, Q_ref  : in  std_logic_vector(31 downto 0);
     L             : in  std_logic_vector(31 downto 0);
     IGBT1         : out std_logic;
```

```
    IGBT2       : out std_logic;
    IGBT3       : out std_logic;
    IGBT4       : out std_logic;
    IGBT5       : out std_logic;
    IGBT6       : out std_logic);

end Top_Inversor;

architecture Top of Top_Inversor is

  component h1_filter
  port(
    clk    : in  std_logic;
    reset  : in  std_logic;
    H1_in  : in  real;
    H1_out : out real);
  end component;

  component abc_dqo
  port(
    clk    : in  std_logic;
    reset  : in  std_logic;
    a      : in  real;
    b      : in  real;
    c      : in  real;
    w      : in  real;
    d      : out real;
    q      : out real);
  end component;

  component ruta_datos1
  port(
    clk      : in  std_logic;
    reset    : in  std_logic;
    Id       : in  real;
    Iq       : in  real;
    Vd       : in  real;
    Vq       : in  real;
    Pref     : in  real;
    Qref     : in  real;
    Id_sal   : out real;
    Iq_sal   : out real;
    Id_fil   : out real;
    Iq_fil   : out real);
  end component;

  component ruta_datos2
    port(
    clk      : in  std_logic;
    reset    : in  std_logic;
```

```
  Id_sal    : in  real;
  Iq_sal    : in  real;
  Id_fil    : in  real;
  Iq_fil    : in  real;
  d_dqo_abc : out real;
  q_dqo_abc : out real;
  decoup    : in  real);
end component;

component dqo_abc
  port(
  clk        : in  std_logic;
  reset      : in  std_logic;
  d          : in  real;
  q          : in  real;
  w          : in  real;
  a          : out real;
  b          : out real;
  c          : out real);
 end component;

signal   s_Decoup     : real :=  0.0094205037;--Value obtained from psim
constant c_Decoup     : real :=  0.0094205037;--Value obtained from psim
                                  -- Antes 0.0014954119480036277
signal   s_L          : real :=  0.0;
signal   s_Vdc_sw     : real :=  0.00001;
constant c_Vdc_offset : real :=  0.0;
constant c_RelVdc     : real :=  0.25;

signal   s_reset      : std_logic;
signal   s_clk        : std_logic;
signal   s_Ia_fil     : real := 0.0;
signal   s_Ib_fil     : real := 0.0;
signal   s_Ic_fil     : real := 0.0;
signal   s_Ia         : real := 0.0;
signal   s_Ib         : real := 0.0;
signal   s_Ic         : real := 0.0;
signal   s_Va         : real := 0.0;
signal   s_Vb         : real := 0.0;
signal   s_Vc         : real := 0.0;
signal   s_Id         : real := 0.0;
signal   s_Iq         : real := 0.0;
signal   s_Vd         : real := 0.0;
signal   s_Vq         : real := 0.0;
signal   s_Pref       : real := 0.0;
signal   s_Qref       : real := 0.0;
signal   s_Id_fil     : real := 0.0;
signal   s_Iq_fil     : real := 0.0;
signal   s_Id_sal     : real := 0.0;
signal   s_Iq_sal     : real := 0.0;
```

```
  signal   s_d_dqo_abc  : real := 0.0;
  signal   s_q_dqo_abc  : real := 0.0;
  signal   s_Vmod_a     : real := 0.0;
  signal   s_Vmod_b     : real := 0.0;
  signal   s_Vmod_c     : real := 0.0;
  signal   s_w          : real := 0.0;
  signal   s_Gen        : real  := -1.0;
signal   s_wt: real := 0.0;
signal   s_wt_real : real := 0.0;
signal   s_wt2pi       : integer;
constant pi           : real := 3.1415926;
signal s_time : time;

  begin
  s_clk    <= clk;
  s_reset  <= reset;
  uut1: h1_filter
    port map (
      clk       => s_clk,
      reset     => s_reset,
      H1_in     => s_Ia,
      H1_out    => s_Ia_fil);

  uut2: h1_filter
    port map (
      clk       => s_clk,
      reset     => s_reset,
      H1_in     => s_Ib,
      H1_out    => s_Ib_fil);

  uut3: h1_filter
    port map (
      clk       => s_clk,
      reset     => s_reset,
      H1_in     => s_Ic,
      H1_out    => s_Ic_fil);

  uut4: abc_dqo
    port map (
      clk       => s_clk,
      reset     => s_reset,
      a         => s_Ia_fil,
      b         => s_Ib_fil,
      c         => s_Ic_fil,
--      w         => s_w,
      w         => s_wt,
      d         => s_Id,
      q         => s_Iq);

  uut5: abc_dqo
```

```
  port map (
    clk      => s_clk,
    reset    => s_reset,
    a        => s_Va,
    b        => s_Vb,
    c        => s_Vc,
    w        => s_wt,
    d        => s_Vd,
    q        => s_Vq);

uut6: ruta_datos1
  port map (
    clk      => s_clk,
    reset    => s_reset,
    Id       => s_Id,
    Iq       => s_Iq,
    Vd       => s_Vd,
    Vq       => s_Vq,
    Pref     => s_Pref,
    Qref     => s_Qref,
    Id_sal   => s_Id_sal,
    Iq_sal   => s_Iq_sal,
    Id_fil   => s_Id_fil,
    Iq_fil   => s_Iq_fil);

uut7: ruta_datos2
  port map (
    clk      => s_clk,
    reset    => s_reset,
    Id_sal   => s_Id_sal,
    Iq_sal   => s_Iq_sal,
    Id_fil   => s_Id_fil,
    Iq_fil   => s_Iq_fil,
    d_dqo_abc => s_d_dqo_abc,
    q_dqo_abc => s_q_dqo_abc,
    decoup   => s_Decoup);

uut8: dqo_abc
  port map (
    clk      => s_clk,
    reset    => s_reset,
    d        => s_d_dqo_abc,
    q        => s_q_dqo_abc,
    w        => s_wt,
    a        => s_Vmod_a,
    b        => s_Vmod_b,
    c        => s_Vmod_c);

-- Generacion de Decoup
  s_Decoup <= (s_L*s_w)/(s_Vdc_sw);
```

```
P_WT: process(reset, clk)
begin
  if reset='1' then
    s_wt <= 0.0;
  elsif clk'event and clk='1' then
    s_wt <= s_wt + s_w*0.000008;
    if s_wt > 6.283185307179586 then
      s_wt <= s_wt - 6.283185307179586476;
    end if;
  end if;
end process;
P_BIEST: process(reset, clk)
begin
  if reset='1' then
    s_Ia        <= 0.0001;
    s_Ib        <= 0.0001;
    s_Ic        <= 0.0001;
    s_Va        <= 0.0001;
    s_Vb        <= 0.0001;
    s_Vc        <= 0.0001;
    s_w         <= 0.0100;
    s_Pref      <= 0.0001;
    s_Qref      <= 0.0001;
    -- Generacion de Decoup
    s_L         <= 0.0001;
    s_Vdc_sw    <= 0.1000;
  elsif clk'event and clk='1' then
    s_Ia        <= real(conv_integer(Ia));
    s_Ib        <= real(conv_integer(Ib));
    s_Ic        <= real(conv_integer(Ic));
    s_Va        <= real(conv_integer(Va));
    s_Vb        <= real(conv_integer(Vb));
    s_Vc        <= real(conv_integer(Vc));
    s_w         <= real(conv_integer(w))*10000.0/2.0**31;
    s_Pref      <= real(conv_integer(P_ref));
    s_Qref      <= real(conv_integer(Q_ref));
    -- Generacion de Decoup
    s_L       <= real(conv_integer(L))/1000000.0;
    s_Vdc_sw  <= real(conv_integer(Vdc_sw))*16384.0/2.0**12;
  end if;
end process;

CMP: process(s_Vmod_a, s_Vmod_b, s_Vmod_c, s_Gen,clk)
begin
  if s_Vmod_a > s_Gen then
     IGBT1 <= '1';
     IGBT4 <= '0';
  else
     IGBT1 <= '0';
```

```
      IGBT4 <= '1';
   end if;

   if s_Vmod_b > s_Gen then
      IGBT2 <= '1';
      IGBT5 <= '0';
   else
      IGBT2 <= '0';
      IGBT5 <= '1';
   end if;
   if s_Vmod_c > s_Gen then
      IGBT3 <= '1';
      IGBT6 <= '0';
   else
      IGBT3 <= '0';
      IGBT6 <= '1';
   end if;
 end PROCESS;

 GEN:process
   constant period: time := 660 us;
   constant amplitude: real:= 2.0;
   constant resolution: integer := 1000;
 begin
   s_Gen <= -1.0;
   for i in 1 to resolution loop
     s_Gen <= s_Gen+amplitude/real(resolution);
     wait for period/resolution/2;
   end loop;
   for i in 1 to resolution loop
     s_Gen <= s_Gen-amplitude/real(resolution);
     wait for period/resolution/2;
   end loop;
 end process;
end Top;
```

Modcoupler_control.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_signed.all;
use work.modcoupler_pack.all;

-- Control declaration
entity modcoupler_control is
  port(clk, reset : in std_logic;
       inputs : in input_array;
       outputs : out std_logic_vector (N_outputs-1 downto 0) );
```

```
end modcoupler_control;

architecture functional of modcoupler_control is

  component top_inversor
    port(
        clk : in std_logic;
        reset : in std_logic;
        ia: in std_logic_vector( 31 downto 0 );
        ib: in std_logic_vector( 31 downto 0 );
        ic: in std_logic_vector( 31 downto 0 );
        va: in std_logic_vector( 31 downto 0 );
        vb: in std_logic_vector( 31 downto 0 );
        vc: in std_logic_vector( 31 downto 0 );
        vdc_sw: in std_logic_vector( 31 downto 0 );
        w: in std_logic_vector( 31 downto 0 );
        p_ref: in std_logic_vector( 31 downto 0 );
        q_ref: in std_logic_vector( 31 downto 0 );
        l: in std_logic_vector( 31 downto 0 );
        i_gbt1: out std_logic;
        i_gbt2: out std_logic;
        i_gbt3: out std_logic;
        i_gbt4: out std_logic;
        i_gbt5: out std_logic;
        i_gbt6: out std_logic );
  end component;

  signal s_ia : std_logic_vector( 31 downto 0 );
  signal s_ib : std_logic_vector( 31 downto 0 );
  signal s_ic : std_logic_vector( 31 downto 0 );
  signal s_va : std_logic_vector( 31 downto 0 );
  signal s_vb : std_logic_vector( 31 downto 0 );
  signal s_vc : std_logic_vector( 31 downto 0 );
  signal s_vdc_sw : std_logic_vector( 31 downto 0 );
  signal s_w : std_logic_vector( 31 downto 0 );
  signal s_p_ref : std_logic_vector( 31 downto 0 );
  signal s_q_ref : std_logic_vector( 31 downto 0 );
  signal s_l : std_logic_vector( 31 downto 0 );

begin

  s_ia <= conv_std_logic_vector( inputs(0), 32 );
  s_ib <= conv_std_logic_vector( inputs(1), 32 );
  s_ic <= conv_std_logic_vector( inputs(2), 32 );
  s_va <= conv_std_logic_vector( inputs(3), 32 );
  s_vb <= conv_std_logic_vector( inputs(4), 32 );
  s_vc <= conv_std_logic_vector( inputs(5), 32 );
  s_vdc_sw <= conv_std_logic_vector( inputs(6), 32 );
  s_w <= conv_std_logic_vector( inputs(7), 32 );
  s_p_ref <= conv_std_logic_vector( inputs(8), 32 );
```

```
  s_q_ref <= conv_std_logic_vector( inputs(9), 32 );
  s_l <= conv_std_logic_vector( inputs(10), 32 );

  top: top_inversor
    port map (
      clk => clk,
      reset => reset,
      ia => s_ia,
      ib => s_ib,
      ic => s_ic,
      va => s_va,
      vb => s_vb,
      vc => s_vc,
      vdc_sw => s_vdc_sw,
      w => s_w,
      p_ref => s_p_ref,
      q_ref => s_q_ref,
      l => s_l,
      i_gbt1 => outputs(0),
      i_gbt2 => outputs(1),
      i_gbt3 => outputs(2),
      i_gbt4 => outputs(3),
      i_gbt5 => outputs(4),
      i_gbt6 => outputs(5));

end functional;
```

Modcoupler_pack.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package modcoupler_pack is
  -- Next 4 lines are handle by psim dll
  constant pm_ratio : integer := 10;
  constant clk_period: time := 8000000 ps;
  constant N_inputs: integer := 11;
  constant N_outputs: integer := 6;

  -- Next lines are never touched
  subtype int_32 is integer range -2147483648 to 2147483647; --(32 bit integer)
  type input_array is array (0 to N_inputs-1) of int_32;

  component fli_model is
  end component;
end modcoupler_pack;

package body modcoupler_pack is
```

```
end modcoupler_pack;

---------------------------------------
------ foreign language component -----
---------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity fli_model is
end fli_model;

architecture a of fli_model is
  attribute foreign of a : architecture is "initForeign ./modcoupler.sl";
begin
end a;
```

Modcoupler_top.vhd

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use work.modcoupler_pack.all;

entity modcoupler_top is
end modcoupler_top;

architecture testbench of modcoupler_top is

  -- Control declaration
  component modcoupler_control is
    port(clk, reset : in std_logic;
         inputs : in input_array;
         outputs : out std_logic_vector (N_outputs-1 downto 0) );
  end component;

  -- Timing signals
  signal time_s, time_p : integer:=0;

  -- Clock and reset
  signal clk: std_logic :='0';
  signal reset : std_logic;
  signal cuenta: integer := 0;

  -- Comunication signals
  signal inputs : input_array;
  signal outputs : std_logic_vector (N_outputs-1 downto 0);

begin
```

```
 -- User control block
 -- The user should attach the control to this block
 the_control: modcoupler_control
   port map (clk,reset,inputs,outputs);

 -- Clock and reset
 clk <= not clk after clk_period/2;
 reset <= '1', '0' after clk_period*(3/2);

 -- Time sync, each N ts, tp increments 1
--  process (time_s)
--  begin
--    if time_s'event then
--      if cuenta=pm_ratio-1 then
--           time_p<=time_p+1;
--       cuenta<=0;
--      else
--        cuenta<=cuenta+1;
--      end if;
--    end if;
--  end process;

 -- Foreign language model instance
 fli_instance : fli_model;

 end testbench;
```